

An Atlas of Characteristic Zero Representations

by

Simon Jon Nickerson

A thesis submitted to
The University of Birmingham
for the degree of
Doctor of Philosophy

School of Mathematics
The University of Birmingham
January 2006

UNIVERSITY OF
BIRMINGHAM

University of Birmingham Research Archive

e-theses repository

This unpublished thesis/dissertation is copyright of the author and/or third parties. The intellectual property rights of the author or third parties in respect of this work are as defined by The Copyright Designs and Patents Act 1988 or as modified by any successor legislation.

Any use made of information contained in this thesis/dissertation must be in accordance with that legislation and must be properly acknowledged. Further distribution or reproduction in any format is prohibited without the permission of the copyright holder.

Abstract

Motivated by the World Wide Web Atlas of Finite Group Representations and the recent classification of low-dimensional representations of quasisimple groups in cross-characteristic fields by Hiss and Malle, we construct with a computer over 650 representations of finite simple groups. Explicit matrices for these representations are available on the Internet and are included on an attached CD-ROM. Our main tool is a GAP program for decomposing permutation modules. It uses reduction modulo various primes and rational reconstruction to give an acceptable performance.

In addition, we define standard generators for the groups under consideration, and exhibit black box algorithms for finding standard generators and checking whether given elements of the group are standard generators.

To my parents

Acknowledgements

I have benefited greatly from the guidance and support of my supervisor, Professor Robert Wilson. I wish to thank him for his encouragement and enthusiasm in this project. I feel privileged to have been one of his students.

I am indebted to my examiners Professor Derek Holt and Dr Paul Flavell for their detailed reading of the text and for pointing out several improvements.

I thank Dr John Bray who has very helpfully shared his knowledge of computational group theory with me. He has also provided two interesting representations for inclusion here. I also thank Richard Barraclough for helping me with the Monster group programs and Dr Frank Lübeck for helping me with my questions about CHEVIE.

My work has been made greatly easier by the GAP computer algebra system. I thank all the developers for their hard work in producing such a marvellous tool and making it freely available.

I am very grateful to Sophie Whyte, Elizabeth Wharton and Marijke van Gans, with whom I have shared an office for the past three years. I wish to thank them for many useful conversations, for helping with the crossword, and for patiently putting up with my annoying habits. I also thank the School of Mathematics at the University of Birmingham for providing a stimulating environment for research.

I gratefully acknowledge the financial support of the Engineering and Physical Sciences Research Council (EPSRC) which allowed me to carry out this research.

My friends and family have been a great support during the past three years. I am deeply grateful to them all for their love, prayers, patience and provision of good coffee. I regret that I was unable to incorporate some of their excellent suggestions (such as introducing 'purple box algorithms')! I thank my fiancée Yvonne with all my heart for her loving support and forbearance, and for making this past year the happiest of my life. Finally, I wish to record special thanks to my parents for their love, their encouragement and their example to me.

Simon Nickerson
3rd January 2006

Contents

0	Introduction	1
0.1	Representations and the Web Atlas	1
0.2	The Hiss-Malle classification	3
0.3	Computer packages and systems	5
0.4	Outline of this thesis	7
I	Standard generators	10
1	Standard generators: definitions and black box algorithms	11
1.1	Introduction to standard generators	12
1.2	Notation for standard generators	13
1.3	Definitions for standard generators	14
1.4	Structure constants	15
1.5	Method of defining standard generators	17
1.5.1	Defining semi-standard pairs	18
1.5.2	Characterising the semi-standard pairs that generate G	19
1.5.3	Counting equivalence classes of semi-standard pairs	19
1.5.4	Choosing an equivalence class	19
1.6	Black box algorithms	20
1.7	Finders	21
1.7.1	Tame generators	22
1.8	Checkers	24
1.8.1	Dihedral groups and involutions	26
1.8.2	Elements of even order	27
1.8.3	Structure constants	27
1.8.4	Fingerprinting	28
2	The sporadic simple groups	30
2.1	Checkers for the sporadic simple groups	31
2.1.1	Mathieu group M_{11}	31
2.1.2	Mathieu group M_{12}	31
2.1.3	Mathieu group M_{22}	31

2.1.4	Mathieu group M_{23}	33
2.1.5	Mathieu group M_{24}	33
2.1.6	Janko group J_1	33
2.1.7	Janko group J_2	33
2.1.8	Janko group J_3	34
2.1.9	Janko group J_4	34
2.1.10	Conway group Co_3	35
2.1.11	Conway group Co_2	35
2.1.12	Conway group Co_1	36
2.1.13	Fischer group Fi_{22}	36
2.1.14	Fischer group Fi_{23}	36
2.1.15	Fischer group Fi'_{24}	37
2.1.16	Higman-Sims group HS	37
2.1.17	Suzuki group Suz	37
2.1.18	McLaughlin group McL	38
2.1.19	Held group He	38
2.1.20	Rudvalis group Ru	39
2.1.21	O'Nan group O'N	39
2.1.22	Harada-Norton group HN	40
2.1.23	Thompson group Th	40
2.1.24	Lyons group Ly	41
2.1.25	Baby Monster group B	42
2.1.26	Monster group \mathbb{M}	42
2.2	Checkers for the sporadic automorphism groups	45
2.2.1	Mathieu group $M_{12}.2$	45
2.2.2	Mathieu group $M_{22}.2$	46
2.2.3	Higman-Sims group HS.2	46
2.2.4	Janko group $J_2.2$	46
2.2.5	Janko group $J_3.2$	47
2.2.6	McLaughlin group McL.2	47
2.2.7	Suzuki group Suz.2	47
2.2.8	Held group He.2	48
2.2.9	O'Nan group O'N.2	48
2.2.10	Fischer group $Fi_{22}.2$	48
2.2.11	Fischer group Fi_{24}	49
2.2.12	Harada-Norton group HN.2	49
2.3	Testing the representations in the Web Atlas	51
3	The alternating and symmetric groups	53
3.1	Standard generators for S_n	53
3.2	Finders for S_n	56
3.2.1	Finding a transposition	56
3.2.2	Finding an $(n - 1)$ -cycle	57

3.3	Checkers for S_n	60
3.4	Standard generators for A_n	61
3.4.1	Standard generators for $A_n, n \geq 7$ odd	64
3.4.2	Standard generators for $A_n, n \geq 8$ even	66
3.5	Finders for A_n	71
3.5.1	Finding a 3-cycle	72
3.5.2	Finding a $\beta(n)$ -cycle	72
3.6	Checkers for A_n	75
3.6.1	Checking x is a 3-cycle	75
3.6.2	Checking y is an $\beta(n)$ -cycle	76
4	The linear groups $L_2(q)$	78
4.1	Basic facts about $L_2(q)$	78
4.2	Standard generators for $L_2(p)$	80
4.2.1	Standard generators for $L_2(p)$ (prime case)	81
4.2.2	Standard generators for $L_2(2^r)$ (even prime-power case)	85
4.2.3	Standard generators for $L_2(p^r)$ (odd prime-power case)	88
4.3	Black box algorithms for $L_2(q)$	89
5	The symplectic groups $S_4(q)$	91
5.1	Standard generators for $S_4(q), q = p^r, p > 3$ prime	91
5.1.1	Semi-standard pairs	91
5.1.2	Standard pairs	95
5.2	Black box algorithms for $S_4(q), q = p^r, p > 3$ prime	96
5.2.1	Distinguishing the two classes of involutions	97
5.2.2	Finding a 3B element	99
5.3	Other groups $S_4(q)$	101
6	The remaining groups in \mathcal{L}^G	102
6.1	Groups with a readily available character table	102
6.1.1	Linear group $L_3(13)$	104
6.1.2	Linear group $L_4(4)$	106
6.1.3	Linear group $L_4(5)$	107
6.1.4	Linear group $L_5(3)$	108
6.1.5	Unitary group $U_3(9)$	109
6.1.6	Unitary group $U_3(13)$	110
6.1.7	Unitary group $U_3(16)$	110
6.1.8	Unitary group $U_4(4)$	111
6.1.9	Unitary group $U_4(5)$	112
6.1.10	Symplectic group $S_4(8)$	113
6.1.11	Symplectic group $S_4(9)$	114
6.1.12	Symplectic group $S_6(5)$	116
6.1.13	Symplectic group $S_8(3)$	117

6.1.14	Orthogonal group $O_{10}^+(2)$	118
6.2	Groups whose character table is not available	119
6.2.1	Unitary group $U_5(4)$	120
6.2.2	Unitary group $U_6(3)$	121
6.2.3	Unitary group $U_8(2)$	122
6.3	Groups where calculating conjugacy classes is impractical	123
6.3.1	Unitary group $U_9(2)$	123
6.3.2	Symplectic group $S_6(7)$	125
6.3.3	Symplectic group $S_{10}(3)$	126
6.4	Cost of the black box algorithms	129
6.5	Checkers for the remaining groups	130

II Characteristic zero representations 131

7	Known constructions	132
7.1	Representations of A_n	132
7.1.1	The Young system	135
7.2	Representations of $L_2(q)$	136
7.2.1	The cuspidal representations of $L_2(q)$	138
7.3	Weil representations of symplectic groups	139
7.3.1	The method	140
7.4	Other known constructions	140
8	Split-P: a GAP system for splitting permutation modules	142
8.1	Notation	142
8.2	The centralizer algebra \mathcal{A}	145
8.3	Adjacency matrices	145
8.4	The left-regular representation of \mathcal{A}	147
8.5	Eigenvectors of adjacency matrices	148
8.6	Finding eigenvectors of elements of \mathcal{B}	150
8.7	Spinning up	151
8.8	Splitting in \mathbb{C}	152
8.9	Computational problems with the field \mathbb{Q}	153
8.10	Reduction modulo p	155
8.11	Rational reconstruction	156
8.12	Reconstructing scalars	157
8.12.1	Solution sets	157
8.12.2	Reconstruction using the Extended Euclidean Algorithm	158
8.13	Reconstruction of quadratic elements	161
8.13.1	Non-splitting primes	161
8.13.2	Reconstruction	161
8.13.3	Problems with irrational reconstruction	162

8.14	Reconstruction of an action on a submodule	162
8.14.1	Decomposition into linear combinations of basis elements	163
8.14.2	Complexity analysis	163
8.14.3	Applying rational reconstruction	164
8.15	Finding suitable permutation representations	165
8.16	Example session with Split-P	167
9	The 231-dimensional representations of M_{24}	171
9.1	Permutation and tensor product depths	171
9.2	The amalgam	173
9.3	The restrictions of ρ	174
9.3.1	Constructing $\rho _H$	174
9.3.2	Constructing $\rho _K$	174
9.4	Finding the intersection L	175
9.5	Standard basis for standard generators of L	175
9.6	Calculations in G	176
9.7	Completing the amalgam	177
9.8	Finding standard generators	178
10	Miscellaneous techniques	179
10.1	Elementary techniques	179
10.1.1	Restriction	179
10.1.2	Induction	180
10.1.3	Algebraic conjugates	180
10.1.4	Automorphs	180
10.2	Decomposing tensor products	181
10.2.1	Plesken-Souvignier splitting	181
10.2.2	MeatAxe methods	184
10.2.3	Decomposing large modules	185
10.2.4	Floating point rational reconstruction	185
10.3	Dixon's method	186
11	A database of group representations	187
11.1	Information supplied	187
11.2	Checking the data	188
11.2.1	Checker test (semi-presentation)	188
11.2.2	CCL test	189
11.2.3	Speeding up the tests	189
11.3	Scope for further work	190
11.3.1	Missing representations	191
11.3.2	Dealing with irrational representations	191
11.3.3	Better bases	192
11.3.4	Testing the rest of the Web Atlas	192

11.3.5	Larger class of groups	192
11.3.6	Hard cases	193
11.3.7	Extending coverage for the sporadic groups	193
A	Table of representations	195
B	BBOX: a language for black box algorithms	200
B.1	Example: finding an element of order 13 in $L_2(13)$	200
B.2	Example: finding standard generators of Fi_{23}	201
B.3	Language basics	203
B.4	Flow control	203
B.5	Types of identifier	204
B.6	Keywords	205
B.6.1	Black box group commands	205
B.6.2	Oracle commands	205
B.6.3	Jumping and looping commands	206
B.6.4	Counter arithmetic	206
B.6.5	Logical commands	206
B.6.6	Terminating commands	208
B.6.7	Debugging commands	208
B.7	An interpreter for the BBOX language	208
B.7.1	Command: prepareblackbox	209
B.7.2	Command: blackbox	209
C	The attached CD-ROM	211
	List of References	212

List of Tables

1.1	Conjugacy classes in A_7	23
2.1	Standard generators for the sporadic (almost-)simple groups	32
2.2	Fingerprints for Ru	40
3.1	Taming sets for the class \mathcal{T}_n of transpositions in S_n	58
3.2	Words for semi-presentations for S_n	61
3.3	Smallest solutions to equations (3.4.16), (3.4.17)	66
3.4	Smallest solutions to equations (3.4.31), (3.4.32)	68
3.5	Taming sets for the class \mathcal{R}_n of 3-cycles in A_n	73
3.6	Sets Γ_n^\pm for distinguishing $\beta(n)$ -cycles	74
3.7	Words t, u for checking that x is a 3-cycle in (3.6.4)	77
4.1	Characters of $SL_2(q)$, q odd (z is the central element of order 2)	79
4.2	Characters of $SL_2(q)$, q even	80
4.3	Some character values of $L_2(p)$, $p \geq 5$ odd	83
4.4	Extra conditions for $L_2(q)$ (even prime-power case)	88
4.5	Odd prime-power standard generators from the Web Atlas	88
4.6	Extra conditions for $L_2(q)$ (odd prime-power case)	90
5.1	Non-zero contributions for $\zeta_G(2B, 3B, \mathcal{C})$	94
6.1	Finders and checkers	103
6.2	Conjugacy classes in $L_3(13)$	104
6.3	Fingerprints for $(2, 3, 61)$ -pairs of $L_3(13)$	105
6.4	Average cost of running black box algorithms	129
7.1	Young diagrams for representations of S_n splitting in A_n	135
7.2	Representations of $L_2(q)$	137
7.3	Representations from other sources	141
8.1	Cost of Gaussian elimination	155
8.2	Extended Euclidean Algorithm for $N = 4199, m = 900$	160
9.1	Permutation and tensor product depths for M_{24}	172

A.1	Representations available	196
B.1	Predicates for the BBOX language	207

Notation and Conventions

We follow ATLAS [11] conventions for naming groups, conjugacy classes, characters and representations. Composition series for groups are written ‘from the bottom up’. Names of computer programs and systems are set in sans-serif. Computer output and listings are monospaced.

$\beta(n), \gamma(n)$	element orders involved in defining standard generators of A_n : see (3.4.2)
Γ_n^+, Γ_n^-	sets of element orders used to distinguish $(n - 2)$ -cycles from other elements of order $n - 2$ in A_n : see section 3.5.2
Δ, Δ_i	orbits of certain centralizers used when defining standard generators: see section 6.2
$\kappa(n)$	number of semi-standard pairs
Λ_i	an orbit of G on $\Omega \times \Omega$
$\Lambda_i(\alpha)$	$\{\beta \in \Omega : (\alpha, \beta) \in \Lambda_i\}$
$\xi(\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3)$	structure constant for classes $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$
ρ, σ	representations of G
χ	a character of G
Ω	a set on which G acts
$\overline{\Omega}$	the permutation module for the action of G on Ω
ω_0	a distinguished point in Ω

$(a_1, a_2, \dots, a_n)_i$	the value a_i , where i is considered modulo n
$\widehat{(\)}$	the stretching operator: see equation (8.5.1)
$\langle\langle \dots \rangle\rangle$	a semi-presentation: see Definition 1.13)
\mathcal{A}	the centralizer algebra of $\overline{\Omega}$: see section 8.2
\mathcal{B}	left regular representation of \mathcal{A} : see section 8.4
\mathcal{C}	a conjugacy class of G
$\text{ccl}(G)$	the set of all conjugacy classes of G
$\text{denom}(x)$	the denominator of $x \in \mathbb{Q}$
G, H, K, L	finite groups
g, h	elements of G
g^h	$h^{-1}gh$; g conjugated by h
$\text{Irr}(G)$	the set of all irreducible characters of G
$\mathcal{L}, \mathcal{L}_0$	a set of genera from the Hiss-Malle classification: see section 0.2
$\mathcal{L}^G, \mathcal{L}_0^G$	set of groups associated with $\mathcal{L}, \mathcal{L}_0$ respectively
$\text{num}(x)$	the numerator of $x \in \mathbb{Q}$
\mathcal{O}	‘big-Oh’ (for orders of magnitude)
o_i	the required order for $w_i(x, y)$
p	a prime, usually the characteristic for G
q	a prime power, usually the size of the underlying field of G
\mathcal{R}_n	conjugacy class of 3-cycles in A_n
S	generating set for G
std	the standard relations in a semi-presentation
\mathcal{T}_n	conjugacy class of transpositions in S_n
$T(\mathcal{C})$	taming set of a conjugacy class \mathcal{C} : see Definition 1.9
$w_i(x, y)$	a word in x and y used in a definition of standard generators
x, y	elements of G , usually standard generators

Chapter 0

Introduction

0.1 Representations and the Web Atlas

The World Wide Web Atlas of Finite Group Representations [57] is a rapidly expanding on-line collection of data about the small finite simple groups and related groups. At present it includes:

- definitions for standard generators;
- black box algorithms for finding standard generators;
- explicit group automorphisms;
- presentations;
- permutation and matrix representations;
- word programs for finding generators of maximal subgroups; and
- conjugacy class representatives.

The Web Atlas (as it is often called) contains information on all the finite simple groups featured in the ATLAS [11] as well as some larger groups.

Because of such tools as the MeatAxe [36, 41] of Parker, the Web Atlas has fairly good coverage of modular representations. However, the Web Atlas has relatively few representations over fields of characteristic zero. The main reason for this is computational difficulty. Though Parker's methods can be adapted to work with \mathbb{Z} , \mathbb{Q} or its extensions [37, 23], there are theoretical and computational problems which make the endeavour a lot more challenging. In a finite field, splitting up a complicated module can be done providing one is willing to wait long enough. With infinite fields, this is not guaranteed, and one must find different methods. Added to this difficulty is that exact computation in \mathbb{Q} is slow, and computation in extensions of \mathbb{Q} is even slower. However, once we have a representation over \mathbb{Q} , we can reduce it modulo any prime which does not divide the common denominator. A similar operation is possible even when dealing with algebraic extensions of \mathbb{Q} . Thus a single characteristic zero representation leads easily to modular representations in infinitely many characteristics.

In this thesis, we will compute many representations in characteristic zero for inclusion in the Web Atlas. Because of the computational difficulties, we will consider only representations with dimension at most 250. As we will see in the next section, a complete list of such representations for the finite simple groups is now known.

Frequently we will treat a finite group G of Lie type in the same way as we treat the sporadic groups. In other words, we often deliberately forget (or downplay) the fact that G belongs to an infinite family of groups and do not try to use any uniform construction for the representations or generators of G . There is an obvious disadvantage with this approach if one is interested in all representations of all finite simple groups but because we are limiting ourselves to representations of dimension at most 250, we are saved a substantial amount of effort if we treat the groups individually. We can use whichever method is easiest to find each representation, without having to consider any other representations. Deligne and Lusztig [14] provide a uniform treatment for

the groups of Lie type for those who require it.

0.2 The Hiss-Malle classification

Gerhard Hiss and Gunter Malle have classified absolutely irreducible representations $\rho : G \rightarrow \mathrm{GL}_d(k)$ where G is a finite quasisimple group, the dimension d is at most 250 and the characteristic of the field k differs from the defining characteristic of G if G is a group of Lie type [21, 22]. From their work, we can easily extract a classification of the irreducible complex representations of the finite simple groups whose dimension does not exceed 250. Our discussion in this section will be about this ‘derived’ classification.

We will use this (non-standard) definition:

Definition 0.1 *Let $\rho : G \rightarrow \mathrm{GL}_d(\mathbb{C})$ be an irreducible faithful complex representation of a finite group. The genus τ of ρ is a 3-tuple (G, d, i) where G is the group (faithfully) represented by ρ , d is the dimension of ρ and $i \in \{-, \circ, +\}$ is the Frobenius-Schur indicator of ρ .*

In essence, the Hiss-Malle classification consists of a list \mathcal{L} of genera, and the following theorem:

Theorem 0.2 (Hiss-Malle) *Let ρ be an irreducible complex representation of a finite simple group whose dimension is at most 250. Let τ be the genus of ρ . Then $\tau \in \mathcal{L}$.*

The set \mathcal{L} is defined by an explicit list (Table 3 in [21], corrected in [22]) together with some generic examples (Table 2 in [21]) of commonly-occurring genera. The generic examples are:

- (G1) The deleted permutation representation of A_n with genus $(A_n, n-1, +)$
- (G2) The Weil representations of $L_2(q)$ ($q \equiv 1 \pmod{4}$) with genus $(L_2(q), (q+1)/2, +)$
- (G3) The Weil representations of $L_2(q)$ ($q \equiv 3 \pmod{4}$) with genus $(L_2(q), (q-1)/2, \circ)$

- (G4) The cuspidal representations of $L_2(q)$ with genus $(L_2(q), q-1, +)$
- (G5) The Steinberg representation of $L_2(q)$ with genus $(L_2(q), q, +)$
- (G6) The representations of $L_2(q)$ with genus $(L_2(q), q+1, +)$ (induced from linear representations of a Borel subgroup)

We define \mathcal{L}_0 to be the set \mathcal{L} with the generic examples (G1)-(G6) deleted. In this thesis, we will concentrate on constructing representations whose genus is in \mathcal{L}_0 (although we will have something to say about the generic examples too). We miss out the generic examples for two reasons. Firstly, there exist uniform constructions for them in the literature, which are sketched in Chapter 7. Secondly, there are very many of these representations to construct, and there seemed little point in adding (say) the 250-dimensional representation of A_{251} to the Web Atlas. We define \mathcal{L}^G (respectively \mathcal{L}_0^G) to be the set of groups which have one or more genera in \mathcal{L} (respectively \mathcal{L}_0). We have:

$$\begin{aligned}
\mathcal{L}_0^G = \{ & M_{11}, M_{12}, M_{22}, M_{23}, M_{24}, \text{HS}, \text{McL}, \text{Co}_3, \text{Co}_2, \text{J}_2, \text{Suz}, \text{Fi}_{22}, \text{He}, \text{HN}, \\
& \text{Th}, \text{J}_1, \text{J}_3, \text{A}_5, \text{A}_6, \dots, \text{A}_{23}, \text{L}_3(3), \text{L}_3(4), \text{L}_3(5), \text{L}_3(7), \text{L}_3(8), \text{L}_3(9), \\
& \text{L}_3(11), \text{L}_3(13), \text{L}_4(3), \text{L}_4(4), \text{L}_4(5), \text{L}_5(3), \text{L}_6(2), \text{L}_7(2), \text{S}_4(4), \\
& \text{S}_4(5), \text{S}_4(7), \text{S}_4(8), \text{S}_4(9), \text{S}_4(11), \text{S}_4(13), \text{S}_4(17), \text{S}_4(19), \text{S}_6(2), \\
& \text{S}_6(3), \text{S}_6(5), \text{S}_6(7), \text{S}_8(2), \text{S}_8(3), \text{S}_{10}(2), \text{U}_3(3), \text{U}_3(4), \text{U}_3(5), \text{U}_3(7), \quad (0.2.1) \\
& \text{U}_3(8), \text{U}_3(9), \text{U}_3(11), \text{U}_3(13), \text{U}_3(16), \text{U}_4(2), \text{U}_4(3), \text{U}_4(4), \text{U}_4(5), \\
& \text{U}_5(4), \text{U}_6(2), \text{U}_6(3), \text{U}_7(2), \text{U}_8(2), \text{U}_9(2), \text{O}_7(3), \text{O}_8^-(2), \text{O}_8^+(2), \\
& \text{O}_8^-(3), \text{O}_{10}^-(2), \text{O}_{10}^+(2), \text{G}_2(3), \text{G}_2(4), \text{G}_2(5), \text{Sz}(8), \text{Sz}(32), \\
& {}^3D_4(2), {}^3D_4(3), {}^2F_4(2)'\}
\end{aligned}$$

and:

$$\mathcal{L}^G = \mathcal{L}_0^G \cup \{A_{24} \dots A_{251}\} \cup \{L_2(7), \dots L_2(499)\} \quad (0.2.2)$$

Note that the Hiss-Malle classification gives a list of the possible genera that occur, but it does not give information about the actual equivalence classes of representations. ‘Essentially different’ representations can have the same genus.

Example 0.3 *There are 3 equivalence classes of irreducible representations with genus*

$$(HS, 154, +).$$

The representations 154b and 154c are automorphic to each other under an outer automorphism of HS. However, the representation 154a is neither automorphic nor algebraically conjugate to 154b or 154c.

We therefore need to consult the character tables of the groups in question to find out how many equivalence classes of representations of each genus there are.

0.3 Computer packages and systems

In recent years, there have been two main general-purpose systems used for computational group theory, with several smaller systems for more specific calculations.

The GAP computer system [19] originated in RWTH-Aachen and has since moved to St Andrews. This computer system was originally designed for computational group theory, although its more recent versions have support for other mathematical objects (such as rings and Lie algebras).

The Magma computer system [4] is based in Sydney, Australia. Its development is headed by John Cannon. In some ways, Magma is a much more ambitious system than GAP, and it is much less tied to group theory calculations.

Most of the programs we wrote for this thesis were implemented in GAP. There is an obvious advantage to staying with a single computer package where possible. Most of the calculations we performed could just as easily have been performed with Magma. We cite the following reasons for our choice of GAP over Magma:

- GAP is open-source software. If we ever needed to, we would be able to see exactly how a given algorithm was implemented.
- GAP (due to its heritage) has better support for character table calculations than Magma.
- It is usually easier to interrupt a calculation in GAP than in Magma.

On the other hand, Magma is sometimes faster than GAP because many of its algorithms are implemented in C rather than in a higher-level language. Occasionally, we found that the GAP functionality was too slow (in particular when dealing with finite fields) and we had to find some alternative.

Other notable computer systems that we used during this project were:

- CHEVIE [20]. This is a package for Maple [53] for performing character table calculations for ‘generic’ groups of Lie type (*i.e.* calculations which depend only on the Dynkin diagram for the group, not the order q of the finite field).
- The Monster library of Parker and Wilson [31]. This is a set of C routines for performing calculations in the Monster simple group. We make use of these routines in section 2.1.26.
- The Perl programming language [55] has proved invaluable for many ‘clerical’ tasks, especially for the large amount of repetitive work involved in organising our database of representations and presenting them via a Web interface.

The computational work for this thesis was done on the Linux Beowulf cluster funded by EPSRC grant GR/R95265/01. Each node had two AMD Athlon MP 1900+ CPUs with 2GB memory.

0.4 Outline of this thesis

Part I: Standard generators

To specify a representation $\rho : G \rightarrow \mathrm{GL}_d(k)$ of a group G , it suffices to give the images under ρ of a set of generators of G . The Web Atlas approach is to choose a single set of generators for each group (known as the *standard generators*) and to use these generators for all representations of this group. In Part I of this thesis, we will define standard generators for all the groups in \mathcal{L}^G which do not already have them defined. In most cases, we also give *finders* (black box algorithms for finding generators) and *checkers* (black box algorithms for checking whether given elements of a group are standard generators).

In Chapter 1, we will describe what is meant by ‘standard generators’, define some basic terms and give some techniques which will be needed in later chapters to find definitions for generators and the various black box algorithms.

In Chapter 2, we consider the sporadic simple groups and their automorphism groups. Standard generators for these groups have already been defined by Wilson [58], and finders have been available in the Web Atlas for a while. We produce checkers for these groups. This chapter provides the basis of an article published in *Experimental Mathematics* [33].

In Chapter 3, we will consider standard generators for the alternating and symmetric groups. The discussion on symmetric groups is slightly tangential to the overall aim of this thesis (as we are concentrating on simple groups), but it provides an introduction to the harder case of alternating groups.

In Chapter 4, we consider the groups $L_2(q)$, which contribute many representations to \mathcal{L} (although not \mathcal{L}_0). As well as a general definition of standard generators for $L_2(p)$ (for any p prime), we are able to produce definitions for the other $L_2(q)$ in \mathcal{L}^G .

In Chapter 5, we consider the groups $S_4(q)$ in \mathcal{L}^G in a similar fashion.

In Chapter 6, we consider the remaining groups in \mathcal{L}^G which do not have standard generators defined either in the Web Atlas or earlier in this thesis.

Part II: Characteristic zero representations

Once we have standard generators defined, we can produce the representations.

In Chapter 7, we give constructions for the irreducible representations of A_n and $L_2(q)$ and the Weil representations of $S_{2n}(q)$. Some of these representations correspond to genera in $\mathcal{L} \setminus \mathcal{L}_0$, and some are used to construct representations in \mathcal{L}_0 .

In Chapter 8, we describe a system we have produced for GAP called Split-P which decomposes permutation modules. This system makes use of reduction modulo p and rational reconstruction to improve speed. It was used to construct most of the representations in this thesis.

In Chapter 9, we briefly describe amalgamation as a technique for producing complex representations and give a construction of the representations $231a/b$ of M_{24} .

In Chapter 10, we give details of certain miscellaneous techniques which were used to find representations.

Finally in Chapter 11, we describe our database of group representations, including details of how it was checked for accuracy and completeness.

Appendices

Appendix A is a table of representations with genera in \mathcal{L}_0 , tabulated according to whether or not the representation has been constructed and is available on the attached CD-ROM (see Appendix C).

Appendix B is a description of a language **BBOX** for black box algorithms. We used this language to write all the black box algorithms for this thesis. The algorithms themselves are available on the CD-ROM, as well as a **BBOX** interpreter for **GAP**.

Appendix C is a note on how to use the attached CD-ROM.

Part I

Standard generators

Chapter 1

Standard generators: definitions and black box algorithms

A representation $\rho : G \rightarrow \mathrm{GL}_d(\mathbb{C})$ of a group G can be specified by giving the images $\rho(g_1), \dots, \rho(g_m)$ of a set of generators $S = \{g_1, \dots, g_m\}$. This is the approach the Web Atlas takes, and will be the approach we take. Thus before we can give any representations, we must first choose the set $S \subset G$ of generators. We will use *standard generators* as defined by Wilson [58]. For many groups (for example, all the sporadic groups), standard generators have already been defined and their definitions appear in the Web Atlas. For the remaining groups, we must define them ourselves. This is our task for Part I of this thesis.

In this chapter, we will define what it means to give a ‘definition of standard generators’ for a group and describe two types of black box algorithm which respectively find and check standard generators. We will also describe the methods we will use in later chapters to give definitions of standard generators.

1.1 Introduction to standard generators

Let G be a group. In general there are many ways to generate G , but for the sake of uniformity and to make it easy to verify calculations and transfer results between different representations, we wish to find an m -tuple of generators that can be characterised up to automorphisms of G using properties that do not depend on any particular representation of G . Our characterisation of this m -tuple will be called a *definition of standard generators*, and any m -tuple satisfying the characterisation will be called an *m -tuple of standard generators*. Before giving a formal definition, we give an example:

Example 1.1 Let $G = A_5$. The group G can be generated by the permutations $x = (1,2)(3,4)$ and $y = (2,4,5)$. We can characterise the pair (x, y) up to automorphisms of G (i.e. up to conjugacy in S_5) by requiring x to have order 2, y to have order 3 and xy to have order 5. This is in fact the definition for standard generators of A_5 in the Web Atlas [57].

Definition 1.2 Let G be a finite group.¹ A definition of standard generators for G is a sequence $P_1(x_1, \dots, x_m), \dots, P_r(x_1, \dots, x_m)$ of predicates in the theory of G such that:

(SG1) There exist $x_1, \dots, x_m \in G$ which satisfy all the predicates P_i ($1 \leq i \leq r$).

(SG2) If $G_1 \cong G$ and $y_1, \dots, y_m \in G_1$ satisfy each of the predicates P_i ($1 \leq i \leq r$) then $\langle y_1, \dots, y_m \rangle = G_1$.

(SG3) If additionally $G_2 \cong G$ and $z_1, \dots, z_m \in G_2$ satisfy each of the predicates P_i ($1 \leq i \leq r$) then there is an isomorphism

$$\sigma : G_1 \rightarrow G_2 \tag{1.1.1}$$

induced by $y_i \mapsto z_i$ ($1 \leq i \leq m$).

¹However, note that definitions of standard generators are really about isomorphism classes of groups rather than groups themselves.

Elements (y_1, \dots, y_m) of $G_1 \cong G$ satisfying these predicates are said to be standard generators of G_1 .

To make standard generators useful in practical situations, the characterising properties P_i are chosen so that an m -tuple with these properties is easy to find in any representation of G . We will justify this by giving a *black box algorithm* (see section 1.6) which outputs a set of standard generators after a small number of operations. Note that ‘small’ in this context is a loosely-defined term, and we make no assertion about minimality.

The main virtue of standard generators is that they allow implicit isomorphisms of groups to become explicit, and thus make it easier to transfer calculations between different representations. If G and H are isomorphic groups, then we can construct an explicit isomorphism $G \rightarrow H$ by finding m -tuples (g_1, g_2, \dots, g_m) and (h_1, h_2, \dots, h_m) of standard generators for G and H , and extending the map $g_i \mapsto h_i$ ($1 \leq i \leq m$) to a group homomorphism $G \rightarrow H$. By this mechanism, we can transfer a calculation which is difficult in one representation, such as the determination of an element’s conjugacy class, to a different representation where the calculation is easier. We can even transfer the calculation to several different representations and combine the results.

1.2 Notation for standard generators

The set up described above is more general than we need, as all the groups we will be dealing with can be 2-generated. Thus we will always choose $m = 2$, and the standard generators of a group G will be called x and y (rather than g_1 and g_2).² We will also use the notations $w_i(x, y)$, \mathcal{C}_i and o_i in the following way: standard generators for G are x and y such that $x \in \mathcal{C}_1$, $y \in \mathcal{C}_2$ and the words $w_1(x, y) \dots w_t(x, y)$ have orders o_1, \dots, o_t

²The Web Atlas uses a and b for simple groups, c and d (and later letters) for almost-simple groups, and capital letters for quasi-simple groups. We do not adopt this notation here.

respectively, where:

$$w_1(x, y) = x \tag{1.2.1}$$

$$w_2(x, y) = y \tag{1.2.2}$$

$$w_3(x, y) = xy \tag{1.2.3}$$

Example 1.3 *Standard generators of the Fischer group Fi_{22} are x and y where x is in class $2A$, y has order 13, xy has order 11 and $(xy)^3xy^2xy(xy^2)^2$ has order 12 [57, 58].*

Thus we have $\mathcal{C}_1 = 2A$, $\mathcal{C}_2 = 13A$, $(o_1, o_2, o_3, o_4) = (2, 13, 11, 12)$ and:

$$w_4(x, y) = (xy)^3xy^2xy(xy^2)^2$$

1.3 Definitions for standard generators

The definitions for standard generators of an group $G \in \mathcal{L}^G$ can be found in various locations throughout this thesis:

1. Certain groups had standard generators already defined in the Web Atlas [57]. We used these definitions when they were available.
2. For certain parameterised families of groups we try to provide a reasonably uniform definition of standard generators. The following families of groups are considered:
 - alternating groups A_n : chapter 3
 - linear groups $L_2(p^r)$, $p \geq 5$ prime: chapter 4
 - symplectic groups $S_4(p^r)$, $p > 2$ prime: chapter 5

We do not provide completely uniform definitions of standard generators for all the simple groups with a given Dynkin diagram, because we need to encode information about the underlying field, which is difficult to do for general Galois fields $\text{GF}(q)$.

3. The remaining groups in \mathcal{L}^G are dealt with in chapter 6.

Note that when we give a definition of standard generators for a group G , we are making the following assertions:

(SG1) m -tuples with the stated properties exist;

(SG2) any m -tuple with these properties generates G ; and

(SG3) this definition characterises an m -tuple of elements of G up to automorphisms of G .

The proof of these assertions will be implicit in the remarks made before the definition, and may involve a computer calculation or reference to a computed character table. To emphasise the fact that we are doing more than making a definition, we will label each definition by ‘Definition-Theorem’.

1.4 Structure constants

In this section, we introduce *structure constants* [25, 26]. These will be useful when looking for characterisations of standard generators later in this chapter.

Let G be a finite group with complex group algebra $\mathbb{C}G$. Let the conjugacy classes of G be $\mathcal{C}^{(1)}, \mathcal{C}^{(2)}, \dots, \mathcal{C}^{(n)}$ with class representatives g_1, g_2, \dots, g_n respectively. Then we can define the *class sums* $K_i \in \mathbb{C}G$ for $1 \leq i \leq n$ by:

$$K_i = \sum_{g \in \mathcal{C}_i} g \tag{1.4.1}$$

The class sums form a basis of $Z(\mathbb{C}G)$, a subalgebra of $\mathbb{C}G$. We therefore have a rule for multiplication in $Z(\mathbb{C}G)$:

$$K_i K_j = \sum_{k=1}^n c_{ijk} K_k \quad (1.4.2)$$

for some constants c_{ijk} with $1 \leq i, j, k \leq n$. Now K_i is central in $\mathbb{C}G$, so by Schur's lemma, in any irreducible representation, it acts like a scalar. If the irreducible representation has character χ , then K_i acts like $\chi(K_i)/\chi(1)$. Then by summing over irreducible characters, we get:

$$\sum_{\chi \in \text{Irr}(G)} \frac{\chi(g_i)\chi(g_j)}{\chi(1)} |\mathcal{C}^{(i)}| |\mathcal{C}^{(j)}| = \sum_{k=1}^n c_{ijk} |\mathcal{C}^{(k)}| \sum_{\chi \in \text{Irr}(G)} \chi(g_k) \quad (1.4.3)$$

Then by column orthogonality, we obtain:

$$c_{ijk} = \frac{|\mathcal{C}^{(i)}| |\mathcal{C}^{(j)}|}{|G|} \sum_{\chi \in \text{Irr}(G)} \frac{\chi(g_i)\chi(g_j)\overline{\chi(g_k)}}{\chi(1)} \quad (1.4.4)$$

The constant c_{ijk} can be interpreted as the number of pairs (h_i, h_j) with $h_i \in \mathcal{C}^{(i)}$, $h_j \in \mathcal{C}^{(j)}$ such that $h_i h_j = g_k$ for our fixed $g_k \in \mathcal{C}^{(k)}$. Usually we are more interested in counting conjugacy classes of triples $(h_i, h_j, h_i h_j)$ with $h_i \in \mathcal{C}^{(i)}$, $h_j \in \mathcal{C}^{(j)}$, $h_i h_j \in \mathcal{C}^{(k)}$. To this end we define ξ_{ijk} ($1 \leq i, j, k \leq n$) given by:

$$\xi_{ijk} = \frac{c_{ijk}}{|C_G(g_k)|} \quad (1.4.5)$$

$$= \frac{|\mathcal{C}^{(i)}| |\mathcal{C}^{(j)}| |\mathcal{C}^{(k)}|}{|G|^2} \sum_{\chi \in \text{Irr}(G)} \frac{\chi(g_i)\chi(g_j)\overline{\chi(g_k)}}{\chi(1)} \quad (1.4.6)$$

$$= \frac{|G|}{|C_G(g_i)| |C_G(g_j)| |C_G(g_k)|} \sum_{\chi \in \text{Irr}(G)} \frac{\chi(g_i)\chi(g_j)\overline{\chi(g_k)}}{\chi(1)} \quad (1.4.7)$$

The constants ξ_{ijk} are known as *structure constants*: we have the following equation:

$$\xi_{ijk} = \sum \frac{1}{|C_G(\langle h_i, h_j \rangle)|} \quad (1.4.8)$$

where the sum is over conjugacy classes of triples $(h_i, h_j, h_i h_j)$ with $h_i \in \mathcal{C}^{(i)}$, $h_j \in \mathcal{C}^{(j)}$, $h_i h_j \in \mathcal{C}^{(k)}$.

We will use the notation $\xi(\mathcal{C}, \mathcal{C}', \mathcal{C}'')$ for the structure constant corresponding to conjugacy classes $\mathcal{C}, \mathcal{C}', \mathcal{C}''$.

1.5 Method of defining standard generators

Definition 1.4 Let G be a finite group. A definition of a semi-standard pair for G is a triple of predicates $P_1(x, y)$, $P_2(x, y)$, $P_3(x, y)$ in the theory of G such that:

(SSP1) There exist elements $x, y \in G$ which satisfy the three predicates and which generate G .

(It is not required that all such pairs of elements generate G .)

(SSP2) The predicate $P_1(x, y)$ determines the conjugacy class of x up to automorphisms of G .

(SSP3) The predicate $P_2(x, y)$ determines the conjugacy class of y up to automorphisms of G .

(SSP4) The predicate $P_3(x, y)$ determines $o(xy)$.

A pair of elements of G satisfying these predicates is said to be a semi-standard pair for G .

We subdivide the problem of defining standard generators of G as follows:

1. Give a definition of a semi-standard pair for G .
2. Characterise the semi-standard pairs that generate G .
3. Count the number of automorphism classes of semi-standard pairs.
4. Find a condition which chooses a unique equivalence class of semi-standard pairs, and use this to give the definition of standard generators.

1.5.1 Defining semi-standard pairs

A semi-standard pair (x, y) for G is our starting point for standard generators of G . We specify the G -conjugacy classes of x and y and give the order of xy .

Notation 1.5 Let $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ be subsets of G ; usually conjugacy classes or unions thereof.

- A $(\mathcal{C}_1, \mathcal{C}_2)$ -pair is a pair (x, y) with $x \in \mathcal{C}_1, y \in \mathcal{C}_2$.
- A $(\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3)$ -pair is a pair (x, y) with $x \in \mathcal{C}_1, y \in \mathcal{C}_2$ such that $xy \in \mathcal{C}_3$.
- If X is any property applicable to pairs of elements of G , we say G is X -generated if there exists an X -pair (x, y) such that $G = \langle x, y \rangle$.
- The subset \mathcal{C}_3 is said to be a target for $(\mathcal{C}_1, \mathcal{C}_2)$ if G is $(\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3)$ -generated.

In this context, a positive integer denotes the union of conjugacy classes of elements of that order.

The first step to defining semi-standard pairs is to choose the conjugacy classes of x and y . It is important that these classes are easy to find, but also that it is easy to find the right conjugacy class of pairs (so $|C_G(x)|$ and $|C_G(y)|$ need to be large). Usually, the easiest way to achieve this is to make x and y elements of low order which can be found by powering up elements of higher order (and which are contained in larger conjugacy classes). Because standard generators should be easy to find in any representation, we avoid specifying conjugacy classes which are hard to distinguish from others.

Once we have chosen the classes \mathcal{C}_1 and \mathcal{C}_2 of x and y , we investigate the targets k (in most cases, k is an integer denoting an order rather than a particular conjugacy class). A complete list of possible classes for xy can be found by looking at the structure constants $\zeta(\mathcal{C}_1, \mathcal{C}_2, -)$, but not all these possibilities are targets. We can get a partial list of targets by choosing $(\mathcal{C}_1, \mathcal{C}_2)$ -pairs (x, y) at random, throwing away any which do not generate G , and looking at the order of xy .

1.5.2 Characterising the semi-standard pairs that generate G

Let (x, y) be a semi-standard pair for G .

If G is to be thought of as belonging to an family of groups being considered together (e.g. G is an alternating group or G is $L_2(p)$ for p prime) then we can examine the set of maximal subgroups of G , and see whether x and y can ever lie in one of these subgroups. Alternatively, we can try to show that $\langle x, y \rangle$ contains a known set of generators of G .

If G is not being treated as part of a larger family of groups, and G has a reasonably small permutation representation, then perhaps the easiest way of proving that semi-standard pairs generate G is to find a representative of each class of semi-standard pairs, and checking computationally whether or not the whole of G is generated.

1.5.3 Counting equivalence classes of semi-standard pairs

We can count equivalence classes of semi-standard pairs using structure constants. By our work in the previous section, we should know the subgroups generated by each type of semi-standard pair. The structure constants $\zeta_{\text{Aut}(G)}(\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3)$ have a contribution of $1/|\text{C}_{\text{Aut}(G)}(\langle x, y \rangle)|$ for every automorphism class of pair (x, y) with $x \in \mathcal{C}_1$, $y \in \mathcal{C}_2$, $xy \in \mathcal{C}_3$.³ If we only specified the order of xy in the definition of semi-standard pair, then we may need to look at a number of classes \mathcal{C}_3 .

1.5.4 Choosing an equivalence class

Suppose we have calculated that there are ℓ automorphism classes of semi-standard pairs— that is, ℓ automorphism classes for a fixed triple $(\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3)$. Often ℓ is greater than 1, so we have not yet specified our pair up to automorphisms of G . To complete the specification, we list semi-standard pairs by random searching, recording an in-

³The classes \mathcal{C}_i are $\text{Aut}(G)$ -conjugacy classes.

variant (the *fingerprint*) for each one. When we have a list of ℓ pairs, all with different fingerprints, we know we have a complete list of representatives; one semi-standard pair from each equivalence class.

Fingerprints have been used for various purposes before: see Parker [36] and Wilson [58]. In our context, a fingerprint is a list of element orders for various words in x and y . The exact nature of the fingerprint depends on the orders of x and y . For example, if x has order 2 and y has order 3, we would record the orders of xy , $xyxy^2$, $xyxyxy^2$, $xyxyxyxy^2$, $xyxyxy^2xy^2$, and so on. We would not (for example) record xy^2 , because it must have the same order as xy . If we do not find ℓ distinct fingerprints after a reasonable amount of searching, then we increase the number of words in the fingerprint and try again.

Once we have ℓ distinct fingerprints, we examine the list and pick out one or more *supplementary conditions* $w_i(x, y)$, $4 \leq i \leq t$ which specify a unique class in the list. Certainly the whole fingerprint would suffice, but we would like our definition of standard generators to be short. In most cases, specifying the orders of one or two short words is enough.

1.6 Black box algorithms

The following definition is taken from Seress [46, chapter 2]:

Definition 1.6 *A black box group G is a group whose elements can be coded as strings of length at most N (for some integer N) over a fixed finite alphabet Q (not necessarily in a unique way) in such a way that given strings s, t representing elements $g, h \in G$ we can:*

- *compute a string representing the element gh ;*
- *compute a string representing the element g^{-1} ; and*
- *decide whether the string s represents the identity of G .*

We do not assume that we can decide whether a string s over the alphabet Q represents an element of G .

Black box groups include permutation groups of finite degree (hereafter simply *permutation groups*) and finite matrix groups over countable fields (hereafter *matrix groups*). Note that all black box groups are finite, because any such can contain at most $\sum_{i=0}^N |Q|^i$ elements. All the representations that we will deal with can be given the structure of a black box group.

In fact, we will need slightly more than is supplied by Definition 1.6. We will require *oracles* for calculating the order of an element and for producing pseudo-random elements with a nearly uniform distribution. We shall assume that all black box groups are equipped with such oracles. Such oracles are available for permutation groups and matrix groups.⁴

A *black box algorithm* is an algorithm which works for any black box group. In Appendix B we describe the simple programming language **BBOX** which can be used to implement black box algorithms, including the ones in this thesis. We have written an interpreter in **GAP** for this language which we used to test the algorithms. It keeps count of the various types of operations performed in the course of the algorithm, which can provide useful information when tuning them.

In this thesis, we are interested in two types of black box algorithm: *finders* and *checkers*. These are discussed in the following two sections.

1.7 Finders

Let G be a group with standard generators defined. We are interested in the problem of finding specific standard generators in a group isomorphic to G .

⁴The oracle for calculating orders of elements $g \in \text{GL}_d(q)$ is due to Celler and Leedham-Green [10]. The order of g can be calculated in $\mathcal{O}(d^3 \log q)$ time providing the numbers $q^i - 1$ for $1 \leq i \leq d$ can be factorised.

Definition 1.7 A finder for G is a black box algorithm which, given a black box group H , satisfies the following properties:

- (F1) If H is isomorphic to G , the algorithm must either output a set of standard generators for H or announce that the algorithm ran out of time.
- (F2) The algorithm terminates after at most T_{\max} operations, for some fixed integer T_{\max} .

We will give a finder for each group G that we consider which (on average) terminates after a small number of operations if it is given a black box group of the right isomorphism type. This provides the justification for our statement that standard generators are easy to find in any representation. For practical reasons, we want our finders to terminate after a small number of operations on average (usually smaller than T_{\max}).

Definition 1.8 The cost T of a finder is the average number of times the random element oracle is consulted to find standard generators.

Because we want low-cost finders, we must choose the semi-standard pair with some care. It may be easy to find a semi-standard pair, but in some representations it may be difficult to prove that a candidate pair really is semi-standard; this is usually because we cannot distinguish between two conjugacy classes containing elements of the same order. Condition (F1) is absolute, and we must be certain (not just fairly confident) that any elements output are standard generators.

1.7.1 Tame generators

Certain definitions for standard generators lend themselves to a particularly simple black box algorithm.

Definition 1.9 Let \mathcal{C} be a subset of G whose elements have the same order o (for example, a conjugacy class). We say \mathcal{C} is tame if there exists an integer r such that all elements of order r

\mathcal{C}	$T(\mathcal{C})$	Class type
1A	$\{1, 2, 3, 4, 5, 6, 7\}$	Completely tame
2A	$\{2, 4, 6\}$	Completely tame
3A	$\{6\}$	Tame (but not completely tame)
3B	$\{\}$	Non-tame
4A	$\{4\}$	Completely tame
5A	$\{5\}$	Completely tame
6A	$\{6\}$	Completely tame
7A	$\{\}$	Non-tame
7B	$\{\}$	Non-tame

Table 1.1: Conjugacy classes in A_7

in G power up to \mathcal{C} (and there exists an element of order r in G). The set of all such r is called the taming set $T(\mathcal{C})$ of \mathcal{C} . If the taming set contains o , then \mathcal{C} is completely tame.

Standard generators for a group G are called tame (resp. completely tame) if the classes to which x and y must belong are both tame (resp. completely tame).

Example 1.10 The conjugacy classes in A_7 can be classified in Table 1.1.

The following black box algorithm is a finder for groups with tame standard generators: it is called the *standard finder*.

Algorithm 1.11 (Standard finder) To find tame standard generators for G given by $x \in \mathcal{C}_1$, $y \in \mathcal{C}_2$, xy of order o_3 with other conditions $w_i(x, y) = o_i$, $4 \leq i \leq t$:

1. Find an element of G with order in the set $T(\mathcal{C}_1)$, and power it up to give an element $x \in \mathcal{C}_1$.
2. Find an element of G with order in the set $T(\mathcal{C}_2)$, and power it up to give an element $u \in \mathcal{C}_2$.
3. Find a conjugate y of u such that xy has order m , and the conditions $w_4(x, y), \dots, w_t(x, y)$ are simultaneously satisfied.

4. Return x and y .

To use this black box algorithm, we only need the definition of standard generators and the sets $T(\mathcal{C}_1)$ and $T(\mathcal{C}_2)$. If a generator class \mathcal{C} containing elements of order o is completely tame, we do not even need to specify $T(\mathcal{C})$, as we can simply take:

$$T(\mathcal{C}) = \{on : n \in \mathbb{N}\} \cap \{o(g) : g \in G\} \quad (1.7.1)$$

1.8 Checkers

Given a group G with standard generators defined, it is sometimes useful to be able to say whether a particular pair of elements of G is a pair of standard generators.

Definition 1.12 *A checker for G is a black box algorithm which, given a black box group H and generators x', y' , satisfies the following properties:*

(C1) *If H is isomorphic to G , the algorithm must return ‘true’ or ‘false’ (depending on whether or not (x', y') is a pair of standard generators for H).*

(C2) *If H is isomorphic to a proper subgroup of G , the algorithm must return ‘false’.*

(C3) *The algorithm terminates after at most T_{\max} operations, for some fixed integer T_{\max} .*

We do not require the algorithm to return ‘false’ if x' and y' generate a group which is not isomorphic to a subgroup of G . To require this would effectively mean calculating a presentation for G on its standard generators, which is in general a hard problem.

The checkers that we find will be given in this thesis in the form of *semi-presentations*⁵.

⁵Semi-presentations can easily be turned into checkers in the sense of Definition 1.12, but the resulting checkers are a much more restrictive class of algorithms than general black box algorithms. Black box algorithms obtained from semi-presentations do not include branching or (pseudo-)random group elements. This is never a problem, as we precompute words in the standard generators whenever we need an element which has certain properties.

Definition 1.13 A semi-presentation for G on a pair (x, y) of standard generators of G is a finite k -tuple of words $w_i(x, y)$, $1 \leq i \leq k$ together with a finite k -tuple of positive integers o_i , $1 \leq i \leq k$ such that:

1. $o(w_i(x, y)) = o_i$ for $1 \leq i \leq k$; and
2. Each pair (x', y') of elements of G satisfying

$$o(w_i(x', y')) = o_i \quad (1 \leq i \leq k)$$

is a pair of standard generators.

Semi-presentations are written in angled brackets $\langle\langle \rangle\rangle$, and the notation $G \approx P$ means that P is a semi-presentation for G . For example, the notation:

$$L_2(7) \approx \langle\langle x, y \mid o(x) = 2, o(y) = 3, o(xy) = 7 \rangle\rangle \quad (1.8.1)$$

asserts that a pair of elements $x, y \in L_2(7)$ is a pair of standard generators if and only if $o(x) = 2$, $o(y) = 3$ and $o(xy) = 7$. Usually however, the standard relations (that is, the relations that can easily be deduced from the definitions of standard generators) are abbreviated by '**std**'. Thus we would write:

$$L_2(7) \approx \langle\langle x, y \mid \mathbf{std} \rangle\rangle \quad (1.8.2)$$

Such a semi-presentation is called the *standard semi-presentation*, and the corresponding checker is the *standard checker*.

The standard relations are usually not enough to give a checker. This is because the standard relations do not check the conjugacy classes of x and y , which invariably form part of the definition for standard generators. For the rest of this section, we

will discuss ways of showing that the given candidates for standard generators are in the correct conjugacy classes in a black box group. These techniques will be used in Chapter 2 and elsewhere to find checkers.

1.8.1 Dihedral groups and involutions

It is well known that two involutions in a finite group generate a dihedral group D_{2n} :

$$D_{2n} = \langle a, b \mid a^2 = b^2 = (ab)^n \rangle \quad (1.8.3)$$

If n is odd, then all the involutions of D_{2n} are conjugate to one another. Thus we have the following useful lemma:

Lemma 1.14 *We have:*

- *If a and b are involutions in a group G and ab has odd order, then a and b are in the same conjugacy class.*
- *Suppose \mathcal{C} and \mathcal{C}' are conjugacy classes of involutions and \mathcal{C}'' is a conjugacy class of odd order. Then*

$$\xi_G(\mathcal{C}, \mathcal{C}', \mathcal{C}'') = 0 \quad \text{unless } \mathcal{C} = \mathcal{C}' \quad (1.8.4)$$

Proof. Clear. ■

The typical application of Lemma 1.14 is to establish the conjugacy class of an involution a . We find an involution b which is known to be in a class \mathcal{C} (say by powering up an element of suitable order), and then find a conjugate b^g of b such that ab^g has odd order. Then by the lemma, $a \in \mathcal{C}$. This turns out to be quite useful, as we frequently have $o_1 = 2$.

1.8.2 Elements of even order

If we are fortunate, we may be able to distinguish classes of even order by powering them up to involutions and using Lemma 1.14. Sometimes this does not work (perhaps the classes power up to the same class of involutions), but we may be able to use their centralizers to distinguish them.

Suppose a is an element of order $2n$, \mathcal{C} and \mathcal{C}' are conjugacy classes of even order $2n$, and p is a prime which divides $|C_G(\mathcal{C})|$ but not $|C_G(\mathcal{C}')|$. Suppose further that we know that a is either in \mathcal{C} or \mathcal{C}' , and we wish to prove that it is in \mathcal{C} . We will try to find an element b of order divisible by p which commutes with a ; because of the centralizer orders, this will show that a is in \mathcal{C} .

In general, finding elements in a centralizer by random methods is hard. With elements of even order, however, we have an advantage because involution centralizers are usually easy to find. Our strategy is:

1. Find words in standard generators which generate the involution centralizer $C_G(a^n)$ (or a sufficiently large subgroup thereof).
2. Find words in the generators of $C_G(a^n)$ which commute with a , generating a sufficiently large subgroup of $C_G(a)$.
3. Find an element in $C_G(a)$ with order divisible by p . By the centralizer orders, we must have that a is in \mathcal{C} .

1.8.3 Structure constants

Sometimes structure constants can be used to establish the conjugacy class of an element. A typical case is when we have classes $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_2'$ and \mathcal{C}_3 where \mathcal{C}_2 and \mathcal{C}_2'

contain elements of the same order and:

$$\xi(\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3) > 0$$

$$\xi(\mathcal{C}_1, \mathcal{C}_2', \mathcal{C}_3) = 0$$

Thus if $x \in \mathcal{C}_1$ and $y \in \mathcal{C}_2 \cup \mathcal{C}_2'$, we can establish that $y \in \mathcal{C}_2$ if we can find a conjugate y^g of y such that $xy^g \in \mathcal{C}_3$.

Example 1.15 *The group $G = \text{Fi}_{23}$ treated in section 2.1.14 has standard generators $x \in 2B$ and $y \in 3D$ such that xy has order 28. Suppose we have checked that all the orders are correct, and that we are also able to establish that x is in class $2B$. Then because $\xi_G(2B, 3A, 28) = 0$, we know that y cannot be in $3A$. For the same reason, y cannot be in $3B$ or $3C$. Thus y must be in $3D$, and no further checking is needed.*

1.8.4 Fingerprinting

If the above methods do not work, or the group G being considered is reasonably small, we may be able to write a checker by identifying all the automorphism classes of (o_1, o_2, o_3) -pairs (x, y) in G and finding a condition on (x, y) which identifies the standard pair. This is similar to the way that standard generators are defined (see section 1.5.4), except that there may be many more pairs to consider. The number K of classes of pairs is usually determined by considering structure constants in G and in subgroups of G . We then look for K different fingerprints, and pick out some conditions which uniquely specify the standard pair.

If there are too many classes of (o_1, o_2, o_3) -pairs, or if K is difficult to determine exactly, we may try to classify pairs of a different type (with fewer fingerprints to consider). This may allow us to identify the class of x or y (or possibly both).

Example 1.16 *Let $G = \text{Co}_1$, considered in section 2.1.12. We are able to show easily that $x \in 2B$. To show $y \in 3C$, we could try fingerprinting the $(2, 3, 40)$ pairs, but there are a lot of them, and they are hard to analyse because not all of them generate G . Instead, we find a $2A$ element t and a $(2A, 3, 36)$ pair (t, y^8) and find fingerprints for all such. This enables us to prove $y \in 3C$.*

Chapter 2

The sporadic simple groups

Standard generators for the sporadic groups were defined by Wilson [58] and seekers for these groups have been in the Web Atlas [57] for some time. In this chapter, we will exhibit checkers for the sporadic groups and their automorphism groups. The material in this chapter forms the basis for an article in Experimental Mathematics [33].

Recall from Definition 1.12 that a checker is a black box algorithm which verifies that a set of elements of a group G is a set of standard generators (under the assumption that these elements all lie in G). We described the techniques that we will need in section 1.8.

Throughout the chapter, we will assume that G is a sporadic simple or almost-simple group and that x and y are elements of G which we wish to show are standard generators of G . Moreover, we assume that x and y obey the ‘standard relations’, *i.e.* $o(w_i(x, y)) = o_i$ for $1 \leq i \leq t$. Thus we only need to find suitable relations which prove $x \in \mathcal{C}_1$ and $y \in \mathcal{C}_2$.

As we remarked earlier, all the checkers will be given in the form of semi-presentations.

2.1 Checkers for the sporadic simple groups

For convenience, we reproduce Wilson's definitions of standard generators for the sporadic simple groups [58, 57] in Table 2.1 on page 32.

2.1.1 Mathieu group M_{11}

There are unique conjugacy classes of elements of orders 2 and 4, so the standard relations suffice to give a semi-presentation. We have:

$$M_{11} \approx \langle \langle x, y \mid \mathbf{std} \rangle \rangle.$$

2.1.2 Mathieu group M_{12}

There are 4 automorphism classes of $(2, 3, 11)$ -pairs, only one of which is a $(2B, 3B, 11)$ -pair. We have:

$$M_{12} \approx \langle \langle x, y \mid \mathbf{std}, o(xyxyxy^2) = 6 \rangle \rangle.$$

2.1.3 Mathieu group M_{22}

We need to show that y is in class $4A$ rather than $4B$. There are 2 automorphism classes of $(2, 4A, 11)$ -pairs and 4 classes of $(2, 4B, 11)$ -pairs. We found fingerprints for all of these. We have:

$$M_{22} \approx \langle \langle x, y \mid \mathbf{std}, o([x, y]) = 6 \rangle \rangle.$$

G	\mathcal{C}_1	\mathcal{C}_2	o_3	w_4	o_4
M_{11}	2	4	11	$xyxy^2xy^3$	5
M_{12}	2B	3B	11		
M_{22}	2	4A	11	$xyxy^2$	11
M_{23}	2	4	23	$(xy)^3xy^2xy(xy^2)^2$	8
M_{24}	2B	3A	23	$(xy)^2xy^2xy(xy^2)^2$	4
J_1	2	3	7	$xyxy^2$	19
J_2	2B	3B	7	$xyxy^2$	12
J_3	2	3A	19	$xyxy^2$	9
J_4	2A	4A	37	$xyxy^2$	10
Co_3	3A	4A	14		
Co_2	2A	5A	28		
Co_1	2B	3C	40	$xyxy^2$	6
Fi_{22}	2A	13	11	$(xy)^3xy^2xy(xy^2)^2$	12
Fi_{23}	2B	3D	28		
Fi'_{24}	2A	3E	29	$xyxyxy^2$	33
HS	2A	5A	11		
Suz	2B	3B	13	$xyxy^2$	15
McL	2A	5A	11	$(xy)^3xy^2xy(xy^2)^2$	7
He	2A	7C	17		
Ru	2B	4A	13		
O'N	2	4A	11		
HN	2A	3B	22	$xyxy^2$	5
Th	2	3A	19		
Ly	2	5A	14	$xyxyxy^2$	67
B	2C	3A	55	$(xy)^3xy^2xy(xy^2)^2$	23
IM	2A	3B	29		
$M_{12.2}$	2C	3A	12	$xyxy^2$	11
$M_{22.2}$	2B	4C	11		
$J_{2.2}$	2C	5AB	14		
$J_{3.2}$	2B	3A	24	$xyxy^2$	9
$Fi_{22.2}$	2A	18E	42		
Fi_{24}	2C	8D	29		
HS.2	2C	5C	30		
Suz.2	2C	3B	28		
McL.2	2B	3B	22	$(xy)^3xy^2xy(xy^2)^2$	24
He.2	2B	6C	30		
O'N.2	2B	4A	22		
HN.2	2C	5A	42		

Table 2.1: Standard generators for the sporadic (almost-)simple groups

2.1.4 Mathieu group M_{23}

There are unique conjugacy classes of elements of orders 2 and 4, so the standard relations suffice to give a semi-presentation. We have:

$$M_{23} \approx \langle \langle x, y \mid \mathbf{std} \rangle \rangle.$$

2.1.5 Mathieu group M_{24}

The $(2B, 3B, 23)$ structure constants are entirely accounted for by the subgroup $L_2(23)$, so we only have one fingerprint for this class of pairs. There are 2 fingerprints for $(2B, 3A, 23)$ -pairs (one of which gives standard generators) and 2 fingerprints for $(2A, 3B, 23)$ -pairs. We have:

$$M_{24} \approx \langle \langle x, y \mid \mathbf{std}, o(xyxyxy^2) = 12 \rangle \rangle.$$

2.1.6 Janko group J_1

There are unique conjugacy classes of elements of orders 2 and 3, so the standard relations suffice to give a semi-presentation. We have:

$$J_1 \approx \langle \langle x, y \mid \mathbf{std} \rangle \rangle.$$

2.1.7 Janko group J_2

The only non-zero $(2, 3, 7)$ -structure constants in J_2 are from $(2A, 3B, 7A)$ (which are accounted for $L_2(7)$, a group which contains no elements of order 12) and $(2B, 3B, 7A)$.

Thus the standard relation $o(xyxy^2) = 12$ is sufficient to ensure that the pair is $(2B, 3B)$:

$$J_2 \approx \langle \langle x, y \mid \mathbf{std} \rangle \rangle.$$

2.1.8 Janko group J_3

The $(2A, 3B, 19)$ -structure constant in $J_3.2$ is 5, but there are only 4 fingerprints because two of the automorphism classes of pairs generate $L_2(19)$, which has an outer automorphism not realised in $J_3.2$. There are 2 classes of $(2A, 3A, 19)$ -pairs, one of which gives standard generators. We have:

$$J_3 \approx \langle \langle x, y \mid \mathbf{std}, o(xyxyxy^2) = 17 \rangle \rangle.$$

In fact, the standard relation $o(xyxy^2) = 9$ is superfluous in the above.

2.1.9 Janko group J_4

We use Lemma 1.14 to show that x is a $2A$ -element (we find our reference $2A$ -element by powering up an element of order 24). To show that y is in $4A$, we follow the method in section 1.8.2 and find an element in $C_G(y)$ with order 20 (the centralizers of elements in classes $4B$ and $4C$ have orders not divisible by 5). We have:

$$\begin{aligned} J_4 \approx \langle \langle x, y, (z, c, d, e, f, g) \mid \mathbf{std}, o(z) = 24, o(x(z^{12})^{xy^3xy^3}) = 11, \\ o(g) = 20, o([g, y]) = 1; z := xyxyxy^2, c := xyxy^3xyxy, \\ d := xy^2xy^3xy^2xy, e := c(y^2(y^2)^c)^5, \\ f := d(y^2)(y^2)^d, g := (efe)^3(fe)^4f \rangle \rangle. \end{aligned}$$

2.1.10 Conway group Co_3

The structure constants are quite large for this group, so we try to eliminate as many cases as we can before fingerprinting. Firstly we check that x is not in $3B$ by checking $o(xy^2) = 24$ (all elements of order 4 have their squares in $2A$, so y^2 is in $2A$, and the $(2A, 3B, 24)$ -structure constant is zero). Secondly we show that y is in $4A$ by finding an element w in its centralizer which has order 5. This leaves the single class of $(3A, 4A, 14)$ -pairs and 341 classes of $(3C, 4A, 14)$ -pairs, for which we found fingerprints. It turns out that the relations we added so far eliminate all the $(3C, 4A, 14)$ -pairs. We have:

$$\begin{aligned} \text{Co}_3 \approx \langle \langle x, y, (u, v, w) \mid \mathbf{std}, o(xy^2) = 24, \\ o(w) = 5, o([w, y]) = 1; u := (y^2(y^2)^{xy^2})^3, \\ v := xyxy^3x^2(y^2(y^2)^{xyxy^3x^2})^2, w := (uv^2)^3(uv)^6 \rangle \rangle. \end{aligned}$$

2.1.11 Conway group Co_2

We can show that x is a $2A$ -element by using Lemma 1.14 (where the reference $2A$ -element is obtained by powering up xy , which is known to have order 28).

Structure constants and fingerprinting show that there is a single automorphism class of pairs of type $(2A, 5A, 28)$ corresponding to standard generators and a single class of pairs of type $(2A, 5B, 28)$ generating a subgroup of $N(2A)$. It turns out that we do not need to add any relations to those we have found already to eliminate this second possibility. We have:

$$\text{Co}_2 \approx \langle \langle x, y \mid \mathbf{std}, o(x(xy)^{14}) = 3 \rangle \rangle.$$

2.1.12 Conway group Co_1

We can show that x is a $2B$ -element by using Lemma 1.14 (where the reference $2B$ -element is obtained by powering up an element of order 42). The structure constant $(2B, 3D, 40)$ is quite large, and to show that y is a $3C$ -element, it is easier to look at $(2A, 3, 36)$ -pairs (see Example 1.16 above). We can find a $2A$ -element by powering up xy (which is known to be of order 40). There are only two fingerprints to consider here: one for $3C$ (which generates a subgroup of $N(2A)$ and has centralizer 2 in G) and one for $3D$ (which generates $\text{U}_6(2):3$, with centralizer 1). We have:

$$\begin{aligned} \text{Co}_1 \approx \langle \langle x, y, (z, a, b) \mid \mathbf{std}, o(z) = 42, o(x^{y^2}z^{21}) = 11, o(ab) = 36, \\ o(ab^2abab) = 18; z := xy(xyxy^2)^2, \\ a := (xy)^{20}, b := y^{xyxyxyxy^2} \rangle \rangle. \end{aligned}$$

2.1.13 Fischer group Fi_{22}

All that needs to be checked is that x is a $2A$ -element. We can use Lemma 1.14, taking the 15th power of an element of order 30 as the reference involution. We have:

$$\text{Fi}_{22} \approx \langle \langle x, y, (z) \mid \mathbf{std}, o(z) = 30, o(xz^{15}) = 3; z := xyxy^2xy^2 \rangle \rangle.$$

2.1.14 Fischer group Fi_{23}

The $(2B, \mathcal{C}, 28)$ structure constant is zero for $\mathcal{C} = 3A, 3B, 3C$, so we only need to show that x is a $2B$ -element (see Example 1.15 above). We can do this by using Lemma 1.14,

taking the 14th power of xy (having order 28) as the reference involution. We have:

$$\text{Fi}_{23} \approx \langle \langle x, y \mid \mathbf{std}, o(x^{y^2}(xy)^{14}) = 5 \rangle \rangle.$$

2.1.15 Fischer group Fi'_{24}

The only non-zero structure constant $\zeta(2A, \mathcal{C}, 29)$ for a conjugacy class \mathcal{C} containing elements of order 3 is for $\mathcal{C} = 3E$. Thus it suffices to check that x is in $2A$. We can do this by using Lemma 1.14, taking the 30th power of an element z of order 60 as the reference involution. We have:

$$\text{Fi}'_{24} \approx \langle \langle x, y, (z) \mid \mathbf{std}, o(z) = 60, o(x(z^{30})^{xyxy}) = 5; z := (xy)^6y \rangle \rangle.$$

2.1.16 Higman-Sims group HS

We found representatives of the classes $2A, 2B, 5A, 5B$ and $5C$ and found fingerprints for all the automorphism classes of $(2, 5, 11)$ -pairs. There are 84 in total. We have:

$$\text{HS} \approx \langle \langle x, y \mid \mathbf{std}, o(xy^2) = 10, o(xyxy^2) = 15 \rangle \rangle.$$

2.1.17 Suzuki group Suz

We have to consider $(2A, 3C, 13)$ -pairs (3 fingerprints), $(2B, 3B, 13)$ -pairs (5 fingerprints) and $(2B, 3C, 13)$ -pairs (63 fingerprints). Note that there are 6 automorphism classes of $(2B, 3C, 13)$ -pairs generating $L_2(25)$, but there are only 3 fingerprints for these because $L_2(25)$ has an extra outer automorphism which is not realised in $\text{Aut}(\text{Suz}) \cong \text{Suz}.2$.

Thus the $(2B, 3C, 13)$ structure constant is 66 rather than 63. We have:

$$\text{Suz} \approx \langle \langle x, y \mid \mathbf{std}, o(xyxyxy^2) = 12 \rangle \rangle.$$

2.1.18 McLaughlin group McL

We have to consider $(2A, 5A, 11)$ -pairs and $(2A, 5B, 11)$ -pairs. There are 2 fingerprints for the former type, one of which gives standard generators. We found 52 fingerprints for the latter type: 34 from pairs generating McL, 14 from pairs generating M_{22} , 2 from pairs generating M_{11} and 2 from pairs generating $L_2(11)$. We account for the structure constant:

$$\xi_{\text{McL}.2}(2A, 5B, 11) = 65 = 34 + 14 \times 2 + \frac{1}{2} (2 + 2 \times 2)$$

by observing that:

- the 14 pairs generating M_{22} are counted twice (because M_{22} has an outer automorphism not realised in $\text{McL}.2$);
- the subgroups M_{11} and $L_2(11)$ have centralizer of order 2 in $\text{McL}.2$; and
- the subgroup $L_2(11)$ has an outer automorphism not realised in $\text{McL}.2$.

We have:

$$\text{McL} \approx \langle \langle x, y \mid \mathbf{std}, o(xy^2) = 12 \rangle \rangle.$$

2.1.19 Held group He

We check that x is a $2A$ -element by using Lemma 1.14, taking the 5th power of an element of order 10 as the reference $2A$ -element. We then consider $(2A, 7A/B, 17)$ -pairs (2 fingerprints), the single $(2A, 7C, 17)$ -pair and the $(2A, 7D/E, 17)$ -pairs (28 finger-

prints). It turns out that the relations we have already are enough to prove that y is in $7C$. We have:

$$\text{He} \approx \langle \langle x, y, (z) \mid \mathbf{std}, o(z) = 10, o(xz^5) = 3; z := xy^2xyxy^2xy^2 \rangle \rangle.$$

2.1.20 Rudvalis group Ru

The structure constants $(2, 4, 13)$ for $G = \text{Ru}$ are fairly complicated, as there are a number of different subgroups of G which can be generated in this way. The only such subgroups which have non-trivial centralizer in G are $\text{Sz}(8)$ and $2 \times \text{Sz}(8)$. Each is contained in a maximal subgroup $(2^2 \times \text{Sz}(8)) : 3$, so each is centralized by a subgroup 2^2 . Both $\text{Sz}(8)$ and $2 \times \text{Sz}(8)$ can be $(2, 4, 13)$ -generated in 4 different ways (up to automorphisms), but because the automorphism of order 3 acts simultaneously on 2^2 and $\text{Sz}(8)$, there are 12 automorphism classes of $(2, 4, 13)$ -pairs in Ru generating $2 \times \text{Sz}(8)$ and only 4 such for $\text{Sz}(8)$. This information, together with the structure constants for Ru and its subgroups, tells us how many fingerprints there should be. The details are given in Table 2.2; overall there are 118 fingerprints to find. We have:

$$\text{Ru} \approx \langle \langle x, y \mid \mathbf{std}, o(xy^2) = 14, o(xyxy^2) = 29 \rangle \rangle.$$

2.1.21 O’Nan group O’N

Here it is sufficient to show that y is a $4A$ -element. We can do this by using the method of section 1.8.2: we find an element z of order divisible by 3, 5 or 7 in its centralizer (as $|C_G(4B)| = 2^8$). Indeed, it is only necessary for z to be in the normalizer of the $4A$ -element (*i.e.* the involution centralizer $C_G(2A)$) as the odd-order part of z will then

\mathcal{C}_1	\mathcal{C}_2	Number of fingerprints ($\mathcal{C}_1, \mathcal{C}_2, 13$)	Subgroups arising
2A	4A	5	$L_3(3):2, L_2(25)$
2A	4B	4	$Sz(8)$
2A	4C	7	$Ru, {}^2F_4(2)'$
2A	4D	29	$Ru, L_3(3), {}^2F_4(2)', L_2(25).2$
2B	4A	1	Ru
2B	4B	10	$Ru, L_2(13).2$
2B	4C	32	Ru
2B	4D	30	$Ru, 2 \times Sz(8)$

Table 2.2: Fingerprints for Ru

centralize y . Hence a suitable z is fairly easy to find. We have:

$$O'N \approx \langle \langle x, y, (z) \mid \mathbf{std}, o(z) = 5, [y, z] = 1; z := xyxy(y^2(y^2)^{xyxy})^5 \rangle \rangle.$$

2.1.22 Harada-Norton group HN

We have $\xi(2A, 3A, 22) = 0$, so it is sufficient to show that x is in $2A$. All elements of order 22 power up to class $2A$, and we know by the standard relations that xy has order 22. Thus we can take $(xy)^{11}$ as our reference involution. Now we search for $z \in G$ such that $x[(xy)^{11}]^z$ has odd order. We have:

$$HN \approx \langle \langle x, y \mid \mathbf{std}, o(x[(xy)^{11}]^{xy^2xyxyxyxy^2}) = 5 \rangle \rangle.$$

2.1.23 Thompson group Th

Here it is sufficient to show that y is a $3A$ -element. Observe that:

$$A_4 \cong \langle g, h \mid g^3 = h^3 = (gh)^2 = 1 \rangle \quad (2.1.1)$$

and g is conjugate to h^{-1} in A_4 . Thus we can show y is a $3A$ -element by taking another $3A$ -element v^{-1} (we take the 7th power of an element z of order 21) and then finding an element w such that yv^w has order 2. We have:

$$\begin{aligned} \text{Th} \approx \langle \langle x, y, (z, v, w) \mid \mathbf{std}, o(z) = 21, o(yv^w) = 2; z := (xy)^3y, \\ v := z^7, w := xy^2(xy)^4(xy^2)^2(xy)^2(xy^2)^5(xy)^3 \rangle \rangle. \end{aligned}$$

2.1.24 Lyons group Ly

We must show that y is a $5A$ -element. To reduce the number of fingerprints to search, we instead look for $(3A, 5, 14)$ -pairs (r, s) . We can find a $3A$ -element r by powering up an element of order 42.

We have:

$$\zeta_{\text{Ly}}(3A, 5A, 14) = 1/3 \quad (2.1.2)$$

$$\zeta_{\text{Ly}}(3A, 5B, 14) = 38/3 \quad (2.1.3)$$

These structure constants are entirely accounted for by the maximal subgroup $3 \cdot \text{McL} : 2$. All the $(3A, 5, 14)$ -pairs in $\text{McL} : 2$ generate McL , so all the $(3A, 5, 14)$ -pairs in Ly generate $3 \cdot \text{McL}$, which is centralized by a group of order 3 in Ly . The structure constants (2.1.2) and (2.1.3) then show that there is just 1 fingerprint for $5A$ and 38 for $5B$. We have:

$$\begin{aligned} \text{Ly} \approx \langle \langle x, y, (z, r, s) \mid \mathbf{std}, o(z) = 42, o(rs) = 14, o(rsrs^2) = 30; \\ z := (xy)^5(xy^2)^2, r := z^{14}, s := y^{xyxy^2xyxyxy^2} \rangle \rangle. \end{aligned}$$

2.1.25 Baby Monster group B

To show that x is in $2C$, we use Lemma 1.14, taking the 26th power of an element of order 52 as our reference $2C$ -element.

To show that y is in $3A$, we observe that all $(2A, 3, 8)$ -pairs in B are $(2A, 3A, 8)$ -pairs (as can be seen from the structure constants). We found an element $z \in 2A$ by taking the 19th power of an element of order 38 and then found a conjugate y^g of y such that zy^g has order 8. (Orders 2, 4 or 14 would also have worked.) Hence (z, y^g) is a $(2A, 3, 8)$ -pair, so y must be in $3A$. We have:

$$\begin{aligned} B \approx \langle \langle x, y, (u, v) \mid \mathbf{std}, o(u) = 52, o(xu^{26}) = 35, o(v) = 38, o(v^{19}y^x) = 8; \\ u := (xyxy^2)^2(xy)^2(xyxy^2)^2; \\ v := (xy)^3(xy^2xy)^2xy(xyxy^2)^2xy^2 \rangle \rangle. \end{aligned}$$

2.1.26 Monster group M

The smallest non-trivial representation of M over any field has dimension 196882, and while standard generators of M in the 196882-dimensional representation over $GF(2)$ have been computed, it is prohibitively slow and expensive in terms of storage to perform calculations with such enormous matrices. Instead, we use the computer construction by Linton et al. [31] with an implementation by Parker and Wilson in the C programming language [28]. We will give an overview of the way we use this construction.

The Monster group M acts linearly on a 196882-dimensional vector space V over $GF(2)$. The group is generated by a subgroup:

$$H = \langle A, B, C, D, E \rangle \simeq 3^{1+12} \cdot 2 \cdot \text{Suz} : 2 \quad (2.1.4)$$

and an extra involution T . Among other things, the construction provides:

- facilities for storing and multiplying elements of the subgroup H ;
- a procedure `vecsuz` for computing the action of an element of H on a vector $v \in V$; and
- a procedure `vecT` for computing the action of T on a vector $v \in V$.

We work with elements of \mathbb{M} expressible in the form $g = h_1 T h_2 T \dots (h_i \in H)$ for reasonably short words.¹ We are able to compute the order of an element $g \in \mathbb{M}$ as follows: we choose a non-zero vector $v \in V$ and compute vg , vg^2 , vg^3 and so on, stopping when we find n such that $vg^n = v$. Then n divides (and is probably equal to) the order of g . When looking for words with a given property, we will assume that n is equal to the order. When we wish to prove our results are correct, we use the two-vector method proposed in Linton et al. [31]. Let r and s be elements of \mathbb{M} with orders 71 and 94 respectively, and let v be a random non-zero vector in V . Let:

$$v_1 = \sum_{i=1}^{71} v r^i \tag{2.1.5}$$

$$v_2 = \sum_{i=1}^{47} v s^{2i-1} \tag{2.1.6}$$

We check that v_1 is a non-zero vector and v_2 is a non-zero vector fixed by s^2 but not s . Then by the known information about the maximal subgroups of \mathbb{M} , the stabilizer of v_1 and v_2 is trivial, and so if n is the least integer such that $v_1 g^n = v_1$ and $v_2 g^n = v_2$, then n is the order of g .

¹Because the words we compute can grow to be arbitrarily long, this does not provide a construction of \mathbb{M} as black box group. This is not a practical difficulty, as we will still be able to produce a black box algorithm. If we really want \mathbb{M} as a black box group, we can compute the action of standard generators x and y on a basis of V to give \mathbb{M} as a matrix group, but as we have observed, a black box operation in this construction would be extremely slow.

Before performing any calculations in the computer construction, we came up with the following strategy for finding a semi-presentation:

1. Find standard generators x, y of \mathbb{M} .
2. Find a word c in x and y whose order is in:

$$S = \{34, 38, 50, 54, 62, 68, 94, 104, 110\}$$

and power it up to give an element $d \in 2A$.

3. Find a word e such that $o(xd^e)$ is in:

$$U = \{1, 3, 5\}.$$

This would prove that x is a $2A$ -element, and also that y is not a $3A$ -element (because $o(xy) = 29$).

4. Find a word f such that $o(xy^f)$ is in:

$$V = \{19, 25, 31, 34, 50, 55, 68, 94\}.$$

The structure constants for \mathbb{M} then imply that y is not a $3C$ -element, so it must be a $3B$ -element.

To find standard generators, we powered up a representative of class $4B$ from [1] to give an involution x , and then looked for conjugates y of a $3B$ class representative b from [59] such that $o(xy) = 29$. (We will be able to prove retrospectively that x and y

are in the correct classes.) We used:

$$\begin{aligned}x &= (DC^3D^2CD^2CD^2CD^2CDCDCDC)^2 \\b &= (ABABAB^2AB)^7 \\y &= ((ABABAB^2AB)^7)^{TBC^3BT}\end{aligned}$$

where the elements $A, B, C, D, T \in \mathbb{M}$ are as described in [31].

We then followed the strategy described above. We have:

$$\begin{aligned}\mathbb{M} \approx \langle \langle x, y, (c, d, f) \mid \mathbf{std}, o(c) = 50, o(xd) = 5, o(xy^f) = 34; \\c := (xy)^4(xy^2)^2, d := c^{25}, f := xyxyxyxyxy^2 \rangle \rangle.\end{aligned}\tag{2.1.7}$$

Step 3 turned out to be unnecessary, as we can take $e = 1$.

We wrote a C program `Monword` which uses the Parker/Wilson routines to check the orders in (2.1.7) above. It is included on the CD-ROM (see Appendix C). It takes just over 3 minutes to run on our system.²

2.2 Checkers for the sporadic automorphism groups

In this section, we will give semi-presentations for the 12 groups $\text{Aut}(G)$ where G is a sporadic simple group and $\text{Out}(G) \neq 1$.

2.2.1 Mathieu group $M_{12}.2$

Here we know that xy is an outer element (it has order 12) and y is inner (it has order 3), and so x must be outer, and hence must be a $2C$ -element. There are 3 fingerprints for $(2C, 3A, 12)$ -pairs (corresponding to the 3 conjugacy classes of elements of order

²We compile with `gcc -O3 -funroll-loops`: these compiler optimisations reduce the running time by about 50%.

12) and 7 fingerprints for $(2C, 3B, 12)$ -pairs. We have:

$$M_{12}.2 \approx \langle \langle x, y \mid \mathbf{std}, o((xy)^3xy^2) = 6 \rangle \rangle.$$

The relation $o(xyxy^2) = 11$ becomes redundant when this extra condition is added.

2.2.2 Mathieu group $M_{22}.2$

There are 13 automorphism classes of $(2, 4, 11)$ -pairs to consider, arising from the different combinations of classes of elements of orders 2 and 4. The subgroups generated in this way have trivial centralizer. We have:

$$M_{22}.2 \approx \langle \langle x, y \mid \mathbf{std}, o(xyxy^2) = 10 \rangle \rangle.$$

2.2.3 Higman-Sims group $HS.2$

We found 29 fingerprints for $(2, 5, 30)$ -pairs; the two subgroups generated with a $(2C, 5B, 30)$ -pair are $5 \times S_5$ and $2 \times A_8$; they have centralizers of orders 5 and 2 (respectively) in G . All other subgroups thus generated have trivial centralizers in G . The structure constants then show that there are exactly 29 automorphism classes of $(2, 5, 30)$ -pairs. We have:

$$HS.2 \approx \langle \langle x, y \mid \mathbf{std}, o([x, y]) = 3 \rangle \rangle.$$

2.2.4 Janko group $J_2.2$

There are 5 automorphism classes of $(2, 5, 14)$ -pairs to find: a unique class of $(2C, 5AB, 14)$ -pairs (the standard generators) and 4 classes of $(2C, 5CD, 14)$ -pairs. All these pairs

generate $J_2.2$. We have:

$$J_2.2 \approx \langle \langle x, y \mid \mathbf{std}, o(xy^2) = 24 \rangle \rangle.$$

2.2.5 Janko group $J_3.2$

By considering element orders in J_3 , x must be in class $2B$. There are 8 automorphism classes of $(2B, 3, 24)$ -pairs, 2 of which correspond to class $3A$. All the pairs generate $J_3.2$. We have:

$$J_3.2 \approx \langle \langle x, y \mid \mathbf{std}, o(xyxyxyxy^2) = 9 \rangle \rangle.$$

The standard relation $o(xyxy^2) = 9$ is redundant.

2.2.6 McLaughlin group $McL.2$

The only non-zero $(2, 3, 22)$ structure constants come from $(2B, 3B)$ -pairs, so the elements must be in the correct conjugacy classes. We have:

$$McL.2 \approx \langle \langle x, y \mid \mathbf{std} \rangle \rangle.$$

2.2.7 Suzuki group $Suz.2$

There are 32 automorphism classes of $(2, 3, 28)$ -pairs, 31 of which generate $Suz.2$, and 1 of which (the unique class of $(2C, 3C, 28)$ -pairs) generates the subgroup $S_4 \times L_3(2)$. We have:

$$Suz.2 \approx \langle \langle x, y \mid \mathbf{std}, o(xyxyxy^2xy^2) = 7 \rangle \rangle.$$

2.2.8 Held group He.2

We can show that x is a $2B$ -element by using Lemma 1.14, taking the 12th power of an element of order 24 as our reference involution. This then implies that y must be an outer element of order 6.

To show that y is in $6C$, we will use the method of section 1.8.2 and find an element of order 15 which commutes with y (the classes $6D$ and $6E$ have centralizers whose orders are not divisible by 5). We have:

$$\begin{aligned} \text{He.2} \approx \langle \langle x, y, (z, t) \mid \mathbf{std}, o(z) = 24, o(xz^{12}) = 17, o(t) = 15, o([t, y]) = 1; \\ z := xy^2xy^2xy, t := (y^3(y^3)^x)^4((y^3(y^3)^{xy^2xy})^2) \rangle. \end{aligned}$$

2.2.9 O'Nan group O'N.2

By the orders of y and xy , we know that x is an outer element, so it must be in class $2B$. There are 2 classes containing elements of order 4, and we want to show that y is in $4A$. Because the $(2B, 4B, 22)$ structure constant is rather large, we chose not to find fingerprints. Instead we used the method of section 1.8.2, and found an element z of order 5 which commutes with y . Since $|C(4B)| = 512$ is not divisible by 5, y must be a $4A$ -element. We have:

$$\begin{aligned} \text{O'N.2} \approx \langle \langle x, y, (t, z) \mid \mathbf{std}, o(z) = 5, o([y, z]) = 1; \\ z := (t(y^2(y^2)^t)^7)^2, t := xy^2xyx \rangle. \end{aligned}$$

2.2.10 Fischer group Fi₂₂.2

We show that x is in class $2A$ by using Lemma 1.14, taking our reference $2A$ -element as the 11th power of an element of order 22.

Because x is an inner element and xy has order 42, y must be an outer element, so it is in one of the classes 18E, 18F, 18G and 18H. We can show that it is either 18E or 18F by considering the 9th power map. The element xy has order 42, so $(xy)^{21}$ is a 2D-element. Lemma 1.14 then allows us to show that y^9 is a 2D-element, so y is in class 18E or 18F. This leaves 5 automorphism classes of $(2A, 18E/F, 42)$ -pairs to test, 2 of which generate the subgroup $3 \times U_4(3).2^2$ with a centralizer of order 3 in G . Each fingerprint gives a different value of $o(xy^8)$, but it turns out that the relations added so far already eliminate the possibility that y is in class 18F. We have:

$$\begin{aligned} \text{Fi}_{22}.2 \approx \langle \langle x, y, (z) \mid \mathbf{std}, o(z) = 22, o(xz^{11}) = 3, \\ o((y^9)^{xy^3}(xy)^{21}) = 3; z := xyxy^5xy^4 \rangle \rangle. \end{aligned}$$

2.2.11 Fischer group Fi_{24}

We check that x is a 2C-element by using Lemma 1.14 (taking the 27th power of an element of order 54 as the reference involution).

The $(2C, 8, 29)$ structure constants are zero except for 8D ($\xi = 1$) and 8F ($\xi = 10$). The only maximal subgroup of Fi_{24} containing an element of order 29 is $29 : 28$, so all the $(2C, 8, 29)$ -pairs generate Fi_{24} . Thus we need to find 11 different fingerprints. We have:

$$\text{Fi}_{24} \approx \langle \langle x, y, (z) \mid \mathbf{std}, o(z) = 54, o(xz^{27}) = 3, o(xy^2) = 20; z := xyxy^6 \rangle \rangle.$$

2.2.12 Harada-Norton group HN.2

By considering element orders, we know that y is an inner element and xy is an outer element. Hence x is an outer element of order 2, so it must be in class 2C.

To show that y is in $5A$, we consider $(2A, 5, 22)$ -pairs. We have:

$$\zeta_{\text{HN.2}}(2A, 5A, 22) = 1/4$$

$$\zeta_{\text{HN.2}}(2A, 5B, 22) = 0$$

$$\zeta_{\text{HN.2}}(2A, 5CD, 22) = 1$$

$$\zeta_{\text{HN.2}}(2A, 5E, 22) = 25/4 = 4 + 9/4$$

We claim that there are (respectively) 1, 0, 1 and 13 classes of $(2, 5, 22)$ -pairs for the classes $5A$, $5B$, $5CD$ and $5E$. We can easily find this many fingerprints; we will show that there cannot be any more.

Certainly any $(2A, 5, 22)$ -pair must be contained in HN. The only maximal subgroup of HN to contain elements of order 22 is 2.HS.2, and in fact these elements are contained in 2.HS. Thus any $(2, 5, 22)$ -subgroup of 2.HS.2 has centralizer of order 4 in HN.2, because we have:

$$2.\text{HS} < 4.\text{HS} < \text{HN.2}$$

and any subgroup of HS containing elements of orders 11 and 5 must have trivial centralizer in HS. Thus each class of pairs either contributes $1/4$ (if it is contained in 2.HS.2) or 1 (if it generates HN).

We consider each class in turn:

- The structure constant shows that there is exactly 1 class of $(2A, 5A, 22)$ -pairs.
- There are no $(2A, 5B, 22)$ -pairs, because the structure constant is zero.
- By considering the fusion between 2.HS.2 and HN and the structure constants in 2.HS.2, we know that all $(2A, 5CD, 22)$ -pairs generates HN. So there is only 1 class of $(2A, 5CD, 22)$ -pairs.

- We observe that 4 of the fingerprints for the $(2A, 5E, 22)$ -pairs have element orders in the set $\{9, 19, 21, 25, 35\}$, showing that they cannot be contained in 2.HS.2 and must therefore generate HN. This leaves a contribution of $25/4 - 4 = 9/4$, and there are $13 - 4 = 9$ fingerprints unaccounted for, so there must be exactly 9 classes generating subgroups of 2.HS.2.

After fingerprinting, it turns out that if (a, b) is a $(2A, 5, 22)$ -pair then:

$$b \in 5A \Leftrightarrow o(ab^2(ab)^3) = 22 \quad (2.2.1)$$

We can find a $2A$ -element t by powering up an element z of order 60. Then we can look for $g \in G$ such that (t, y^g) is a $(2A, 5, 22)$ -pair. We then use the criterion in equation (2.2.1) to check that y^g (and hence y) is a $5A$ element. We have:

$$\begin{aligned} \text{HN.2} \approx \langle \langle x, y, (t, z) \mid \mathbf{std}, o(z) = 60, o(ty) = 22, \\ o(ty^2(ty)^3) = 22; z := xy^3(xy)^4, t := z^{30} \rangle \rangle. \end{aligned}$$

2.3 Testing the representations in the Web Atlas

We used our semi-presentations to test the representations of sporadic simple and almost-simple groups given in the Web Atlas. As we expected, the vast majority of the representations satisfied the relevant semi-presentations, but a few mistakes were discovered:

- Matrices purporting to generate a 483-dimensional representation of M_{23} over $\text{GF}(7)$ were included, but they failed to satisfy the semi-presentation. In fact no such representation of M_{23} exists [27].
- One of the 896-dimensional representations of HS over $\text{GF}(4)$ was incorrect, as

the product of the two generators had order exceeding 100.

- Matrices purporting to generate a 104-dimensional representation of He_2 over $\text{GF}(5)$ in fact generated a group of order 30240.
- The 924-dimensional representation of $\text{Fi}_{22}.2$ over $\text{GF}(3)$ had non-standard generators; the second generator given was xy rather than y .

Chapter 3

The alternating and symmetric groups

In this chapter we will define standard generators for the symmetric groups S_n and alternating groups A_n ($n \geq 5$). We will then exhibit finders for these groups for the case $n \leq 25$.

Black box algorithms for constructive recognition of symmetric and alternating groups have also been investigated by Bratus and Pak [5] and Beals et al [2].

3.1 Standard generators for S_n

We begin with the symmetric groups, where the analysis is somewhat simpler. Let $G = S_n$, and let:

$$x = (1, 2), \quad y = (2, 3, 4 \dots n) \tag{3.1.1}$$

We note that x is a transposition, y has order $n - 1$ and xy has order n . We seek to make this the basis for our definition of standard generators for S_n . (This is an extrapolation of the definition of standard generators already given in the Web Atlas for the small symmetric groups.)

Lemma 3.1 *The group S_n is generated by x and y .*

Proof. Let $t_k = (k, k+1)$ for $1 \leq k \leq n-1$ and let $t = (i, j)$ be an arbitrary transposition with $i < j$. The element $u = t_{i+1}t_{i+2} \cdots t_{j-1}$ fixes i and sends $i+1$ to j . Hence $t = t_i^u$. Since G is generated by transpositions t , we have:

$$G = \langle t_k : 1 \leq k \leq n-1 \rangle \quad (3.1.2)$$

The element $z = (yx)^{i-1}$ sends 1 to i and 2 to $i+1$. Hence $t_i = x^z$, and so $G = \langle x, y \rangle$. ■

To characterise these generators, suppose x is a transposition, y an element of order $n-1$ (not necessarily an $(n-1)$ -cycle) such that xy has order n . Without loss of generality (*i.e.* by conjugating by a suitable element of G), we can suppose $x = (1, 2)$. Because $o(y)$ and $o(xy)$ are coprime, each cycle in the disjoint cycle decomposition of y must move 1 or 2. There are just two possibilities:

(S1) 1 and 2 are contained in the same cycle of y .

We can write

$$y = (1, \underbrace{\circ, \dots, \circ}_{r-1}, 2, \underbrace{\bullet, \dots, \bullet}_{s-1}) \quad (3.1.3)$$

for integers $r, s \geq 1$. Then

$$r + s = n - 1 \quad (3.1.4)$$

and so we get:

$$xy = (1, \underbrace{\bullet, \dots, \bullet}_{s-1})(2, \underbrace{\circ, \dots, \circ}_{r-1}) \implies \text{lcm}(r, s) = n \quad (3.1.5)$$

Now r and s must be proper divisors of n , so each is at most $n/2$. But because $r + s = n - 1$, they must take the values $n/2$ and $n/2 - 1$ in some order. Since

these are coprime, (3.1.5) implies

$$n = \frac{n}{2} \left(\frac{n}{2} - 1 \right) \quad (3.1.6)$$

which implies $n = 6$.

(S2) 1 and 2 are contained in different cycles in y .

$$y = (1, \underbrace{\circ, \dots, \circ}_{r-1}) (2, \underbrace{\bullet, \dots, \bullet}_{s-1}) \implies \text{lcm}(r, s) = n - 1 \quad (3.1.7)$$

$$xy = (1, \underbrace{\bullet, \dots, \bullet}_{s-1}) (2, \underbrace{\circ, \dots, \circ}_{r-1}) \implies r + s = n \quad (3.1.8)$$

If r and s were both proper divisors of $n - 1$ then they would be at most $(n - 1)/2$, which contradicts (3.1.8). Thus they must take the values $n - 1$ and 1 in some order, *i.e.* y is an $(n - 1)$ -cycle and xy is an n -cycle.

This analysis proves:

Lemma 3.2 *Let $G = S_n$ for $n \geq 5$, $n \neq 6$. Suppose $x \in G$ is a transposition and $y \in G$ is an element of order $n - 1$ such that xy has order n . Then y is an $(n - 1)$ -cycle and xy is an n -cycle. Moreover, there is an automorphism of G which sends x to $(1, 2)$ and y to $(2, \dots, n)$.*

Example 3.3 *The case $n = 6$ is a genuine exception to the lemma above. Note that we can take $x = (1, 2)$, $y = (1, 3, 2, 4, 5)$ so that $xy = (1, 4, 5)(2, 3)$ has order 6, but x and y clearly generate S_5 rather than S_6 .*

Lemma 3.2 allows us to give the following definition for standard generators of S_n , $n \geq 5$ ($n \neq 6$):

Definition-Theorem 3.4 *Standard generators of S_n for $n \geq 5$, $n \neq 6$ are given by an element x of order 2 (in the smallest conjugacy class of involutions) and an element y of order $n - 1$ such that their product has order n . (The element y is necessarily an $(n - 1)$ -cycle.)*

3.2 Finders for S_n

In the following sections, we will find finders for the small (*i.e.* $5 \leq n \leq 25$) symmetric groups.

Providing we can find a transposition x and an $(n - 1)$ -cycle y' (or elements auto-morphic to them in the case of S_6), we can find standard generators x, y by taking conjugates y of y' until xy has order n . The probability of finding standard generators in any one attempt is:

$$p = \frac{|C_G(x)| \cdot |C_G(y)|}{|G|} = \frac{2(n-2)!(n-1)}{n!} = 2/n \quad (3.2.1)$$

So the problem reduces to that of finding a transposition (section 3.2.1) and finding an $(n - 1)$ -cycle (section 3.2.2).

3.2.1 Finding a transposition

Let \mathcal{T}_n denote the conjugacy class of transpositions in G . The class \mathcal{T}_n is very small (having centralizer of order $2(n - 2)!$) so we try to find an element in it by powering up even-order elements. For $m \in T(\mathcal{T}_n)$, all elements x of order m must contain exactly one transposition in their disjoint cycle decomposition, and no $(2r)$ -cycles for $r > 1$.

Theorem 3.6 below shows that for $n \geq 5$ (and $n \neq 6$), the taming set $T(\mathcal{T}_n)$ is always non-empty. First of all, we need a lemma.

Lemma 3.5 *Any integer greater than 9 can be expressed as a sum of distinct odd primes.*

Proof. This is proved by Dressler [18] using induction and the following result:

$$p_{n+1} < 2p_n - 10 \quad (n > 6) \quad (3.2.2)$$

which is a strengthening of Bertrand's Postulate. ■

Theorem 3.6 *Let $n \geq 5$, $n \neq 6$ be an integer. Then:*

1. *There exists an integer t_n such that all elements of order t_n in S_n power up to give a transposition.*
2. *The taming set $T(\mathcal{T}_n)$ is non-empty.*

Proof. Clearly part 2 follows from part 1. We can take $t_5 = 6$, $t_7 = 10$, $t_8 = 10$, $t_9 = 14$, $t_{10} = 14$ and $t_{11} = 18$. Otherwise, suppose $n > 11$. By Lemma 3.5 we can write:

$$n - 2 = \sum_{i=1}^k p_i \quad (3.2.3)$$

where p_1, \dots, p_k are distinct odd primes.

Let $t_n = 2 \prod p_i$. By (3.2.3), any element of S_n with order t_n is fixed point free and is a product of $k + 1$ disjoint cycles with orders $2, p_1 \dots p_k$ respectively. Hence any element of order t_n in S_n powers up to a transposition. ■

Note that $n = 6$ is a genuine exception to Theorem 3.6. This is because the outer automorphism of S_6 swaps the conjugacy class of transpositions with the conjugacy class of elements with cycle shape 2^3 . We can ensure that we get one of these two classes by powering up any element of order 6.

The taming sets $T(\mathcal{T}_n)$ for small values of n are given in Table 3.1, together with the approximate probabilities of finding an element with order $m \in T(\mathcal{T}_n)$.

3.2.2 Finding an $(n - 1)$ -cycle

The $(n - 1)$ -cycles form a large conjugacy class in S_n , and the probability of finding one in a random search is $1/(n - 1)$. They also form the largest class of elements of order $n - 1$. However, it may not be possible to determine whether a given element of order $n - 1$ is really an $(n - 1)$ -cycle. Unlike transpositions, $(n - 1)$ -cycles cannot be

n	$T(\mathcal{T}_n)$	$p \approx$
5	6	1/6
6	6^a	1/3
7	10	1/10
8	10	1/10
9	14	1/14
10	14, 30	1/10
11	18	1/18
12	18, 42	1/13
13	22	1/22
14	22, 70	1/17
15	26, 70	1/19
16	26, 66, 90	1/15
17	90, 210	1/63
18	78, 110, 126	1/34
19	34, 110, 126	1/22
20	34, 130, 154	1/23
21	38, 130, 154, 330	1/23
22	38, 102, 182, 198	1/21
23	182, 198, 390, 462, 630	1/59
24	114, 170, 234, 630	1/49
25	46, 170, 234, 546, 770	1/29
26	46, 190, 238, 286, 770	1/28
27	50, 190, 238, 286, 510, 910, 990	1/27
28	50, 138, 266, 306, 910, 990, 2310	1/27
29	54, 266, 306, 570, 714, 858, 1170, 1386	1/32
30	54, 150, 230, 342, 374, 1170, 1386, 2730	1/27

^aA 6-element in S_6 powers up to either a 2-cycle or an element of shape 2^3 , but these classes fuse in $\text{Aut}(S_6)$

Table 3.1: Taming sets for the class \mathcal{T}_n of transpositions in S_n

obtained from elements of higher order by powering up. However, by Lemma 3.2 an element of order $n - 1$ which is not in the right class can never give an element of order n when multiplied by a transposition. We can therefore find any element of order $n - 1$ and go on to the conjugating stage, bearing in mind that we will have to backtrack if we do not find standard generators after a reasonable amount of searching. For high values of n , this may be the best way of finding standard generators. If we wish to find a method which does not require backtracking then we must be more careful.

For $n \in \{5, 6, 8, 9, 10, 12, 14, 17, 18, 20, 24\}$ there is a unique conjugacy class of elements of order $n - 1$, so these must be the $(n - 1)$ -cycles. The probability of finding such an element in a random search is $1/(n - 1)$.

For $n \in \{5, 7, 8, 9, 11, 13, 16, 17, 19, 23, 25\}$ there is a unique conjugacy class of elements of order n , so we can work backwards: take an element z' of order n (probability $1/n$) and look for conjugates z of z' such that xz has order $n - 1$. Then $y = xz$ is an $(n - 1)$ -cycle by Lemma 3.2. We have $xy = x^2z = z$, so x and y are standard generators of S_n , and there is no need to conjugate y any further.

This leaves the cases $n \in \{15, 21, 22\}$. In these cases, we can find an $(n - 1)$ -cycle by working upwards from smaller symmetric groups, at the cost of more random searching:

- For $n = 15$, find an element t of order 13 ($p = 1/26$) and find a conjugate t' of it such that xt' has order 14 ($p \approx 1/4$). Then xt' is a 14-cycle.
- For $n = 21$, find an element t of order 19 ($p = 1/38$) and find a conjugate t' of it such that xt' has order 20 ($p \approx 1/6$). Then xt' is a 20-cycle.
- For $n = 22$, find an element t of order 19 ($p = 1/114$) and find a conjugate t' of it such that xt' has order 20 ($p \approx 1/2$). Let $u = xt'$ and find a conjugate u' of u such that xu' has order 21 ($p \approx 1/6$). Then xu' is a 21-cycle.

3.3 Checkers for S_n

A checker for S_n needs to verify that the first generator x is a transposition. If this is done, then by Definition-Theorem 3.4, we only need to check the standard relations $o(x) = n - 1, o(xy) = n$.

If we want a general strategy for checking S_n ($n \geq 7$), we can exploit the fact that S_n is the Weyl group for the Coxeter system of type A_{n-1} .

$$\begin{aligned} S_n \approx \langle \langle x, y, (t_1, \dots, t_{n-1}) \mid \mathbf{std}, \\ o(t_i t_{i+1}) = 3 \quad (1 \leq i \leq n-2), \\ o(t_i t_j) = 2 \quad (1 \leq i, j \leq n-1, |i-j| > 1); \\ t_1 := x, t_{i+1} := t_i^{yx} \quad (1 \leq i \leq n-2) \rangle \rangle \end{aligned} \quad (3.3.1)$$

(The elements t_i satisfy the Coxeter presentation for S_n , so are guaranteed to be transpositions, and hence x is a transposition.)

If we do not require a general solution, then we can use the dihedral trick and the taming sets $T(\mathcal{T}_n)$ which we found earlier to give a shorter semi-presentation. We find a word z in x and y with $o(z) = r$ such that $r \in T(\mathcal{T}_n)$ and $o(xz^{r/2}) = 3$. Thus x and $z^{r/2}$ are conjugate, i.e. x is a transposition:

$$S_n \approx \langle \langle x, y, (z) \mid \mathbf{std}, o(z) = r, o(xz^{r/2}) = 3; z := \text{word in } x \text{ and } y \rangle \rangle \quad (3.3.2)$$

The following lemma gives a possible value of z for a number of small cases.

Lemma 3.7 *If n is odd and $2(n-2) \in T(\mathcal{T}_n)$, then we can take:*

$$z = xy^{n-3}xy = (1, n, n-1, \dots, 4)(2, 3), \quad o(z) = 2(n-2) \quad (3.3.3)$$

n	z	$r = o(z)$	Permutation
17	$xy^2xy^5xy^9xy$	90	$(2, 17)(3, 4, \dots, 11)(12, 13, 14, 15, 16)$
18	$xy^{13}xy^3xy^2$	78	$(1, 3)(2, 7, 8, \dots, 18)(4, 5, 6)$
23	$xy^2xy^{13}xy^7xy$	182	$(2, 23)(3, 4, \dots, 9)(10, 11, \dots, 22)$
24	$xy^{19}xy^3xy^2$	114	$(1, 3)(2, 7, 8, \dots, 24)(4, 5, 6)$

Table 3.2: Words for semi-presentations for S_n

If n is even and $2(n-3) \in T(\mathcal{T}_n)$, then we can take:

$$z = xyxy^{n-3}xy^2 = (1, 3)(4, 5, \dots, n), \quad o(z) = 2(n-3) \quad (3.3.4)$$

Proof. Straightforward calculation. ■

For $7 \leq n \leq 30$, the only cases not covered by Lemma 3.7 are $n \in \{17, 18, 23, 24\}$.

For these cases we can use the words given in Table 3.2.

3.4 Standard generators for A_n

The situation for the alternating groups is more complicated, as our choice of standard generators depends on the parity of n . As before, we extrapolate the general definition of standard generators for A_n from those given in the Web Atlas.

The standard generators defined in the Web Atlas for A_5 and A_6 differ from the usual pattern.¹ The group A_5 has a triangle presentation:

$$A_5 \cong \langle x, y \mid x^2 = y^3 = (xy)^5 = 1 \rangle \quad (3.4.1)$$

and we make this the basis of standard generators for A_5 : standard generators are x of order 2, y of order 3 such that xy has order 5 (e.g. $x = (1, 2)(3, 4)$, $y = (2, 4, 5)$).

¹This is because there are isomorphisms $A_5 \cong L_2(4) \cong L_2(5)$ and $A_6 \cong L_2(9)$, and groups $L_2(q)$ have an involution as their first standard generator. There are also complications because A_6 has a larger outer automorphism group than the rest of the alternating groups.

Standard generators of the alternating group A_6 are x of order 2, y of order 4 such that xy has order 5 (e.g. $x = (1,2)(3,4)$, $y = (2,4,5,6)$).

From now on, we assume $n \geq 7$. Define:

$$\beta(n) = \begin{cases} n-2 & \text{if } n \text{ is odd} \\ n-1 & \text{if } n \text{ is even} \end{cases} \quad \gamma(n) = \begin{cases} n & \text{if } n \text{ is odd} \\ n-2 & \text{if } n \text{ is even} \end{cases} \quad (3.4.2)$$

Our choice for standard generators of A_n is:

$$x = (1,2,3); \quad y = \begin{cases} (3,4,\dots,n) & \text{if } n \text{ is odd} \\ (2,3,\dots,n) & \text{if } n \text{ is even} \end{cases} \quad (3.4.3)$$

i.e. a 3-cycle and an $\beta(n)$ -cycle whose product has order $\gamma(n)$. In the following sections we will try to prove a result in the spirit of Lemma 3.2 for the alternating groups. Unfortunately, there are exceptional cases which are hard to classify.

Suppose x is a 3-cycle and y is an element of order $\beta(n)$ (not necessarily a $\beta(n)$ -cycle) whose product has order $\gamma(n)$. Without loss of generality, we can assume $x = (1,2,3)$. Any non-trivial cycle of y which does not involve 1, 2 or 3 is unchanged in xy . But $\beta(n)$ and $\gamma(n)$ are coprime, so y cannot have any such cycles.

Hence the non-trivial cycles of y can be found among the cycles of y containing 1, 2 and 3. If we consider the cycles of y containing 1, 2 and 3, there are essentially 4 possibilities. In each case we get two number theoretic conditions on n .

(A1) All 3 points are contained in a single cycle in the order 1, 2, 3.

$$y = (1, \underbrace{\circ, \dots, \circ}_{r-1}, 2, \underbrace{\bullet, \dots, \bullet}_{s-1}, 3, \underbrace{\star, \dots, \star}_{t-1}) \implies r + s + t = \beta(n) \quad (3.4.4)$$

$$xy = (1, \underbrace{\bullet, \dots, \bullet}_{s-1}, 3, \underbrace{\circ, \dots, \circ}_{r-1}, 2, \underbrace{\star, \dots, \star}_{t-1}) \implies r + s + t = \gamma(n) \quad (3.4.5)$$

Since $\beta(n) \neq \gamma(n)$, this case does not occur.

(A2) All 3 points are contained in a single cycle in the order 1, 3, 2.

$$y = (1, \underbrace{\circ, \dots, \circ}_{r-1}, 3, \underbrace{\star, \dots, \star}_{t-1}, 2, \underbrace{\bullet, \dots, \bullet}_{s-1}) \implies r + s + t = \beta(n) \quad (3.4.6)$$

$$xy = (1, \underbrace{\bullet, \dots, \bullet}_{s-1}, 2, \underbrace{\star, \dots, \star}_{t-1}, 3, \underbrace{\circ, \dots, \circ}_{r-1}) \implies \text{lcm}(r, s, t) = \gamma(n) \quad (3.4.7)$$

(A3) The 3 points are contained in two cycles (2 points in one, 1 in the other).

$$y = (1, \underbrace{\circ, \dots, \circ}_{r-1}, 2, \underbrace{\bullet, \dots, \bullet}_{s-1}, 3, \underbrace{\star, \dots, \star}_{t-1}) \implies \text{lcm}(r + s, t) = \beta(n) \quad (3.4.8)$$

$$xy = (1, \underbrace{\bullet, \dots, \bullet}_{s-1}, 2, \underbrace{\star, \dots, \star}_{t-1}, 3, \underbrace{\circ, \dots, \circ}_{r-1}) \implies \text{lcm}(s, r + t) = \gamma(n) \quad (3.4.9)$$

(A4) The 3 points are contained in three cycles (1 point in each).

$$y = (1, \underbrace{\circ, \dots, \circ}_{r-1}, 2, \underbrace{\bullet, \dots, \bullet}_{s-1}, 3, \underbrace{\star, \dots, \star}_{t-1}) \implies \text{lcm}(r, s, t) = \beta(n) \quad (3.4.10)$$

$$xy = (1, \underbrace{\bullet, \dots, \bullet}_{s-1}, 2, \underbrace{\star, \dots, \star}_{t-1}, 3, \underbrace{\circ, \dots, \circ}_{r-1}) \implies r + s + t = \gamma(n) \quad (3.4.11)$$

3.4.1 Standard generators for A_n , $n \geq 7$ odd

For this section, we suppose n is odd and we investigate each of the cases in turn.

(A1) All 3 points are contained in a single cycle in the order 1, 2, 3.

We have already remarked that this is impossible.

(A2) All 3 points are contained in a single cycle in the order 1, 3, 2.

$$r + s + t = n - 2 \tag{3.4.12}$$

$$\text{lcm}(r, s, t) = n \tag{3.4.13}$$

By (3.4.12), r , s and t must be proper divisors of n , so

$$\max(r, s, t) \leq n/3 \tag{3.4.14}$$

Because r , s and t are odd, this forces r , s and t to take the values $n/3$, $n/3$ and $n/3 - 2$ (in some order). Because n is odd, $n/3$ and $n/3 - 2$ are coprime, (3.4.13) implies:

$$n = n/3 \times (n/3 - 2) \tag{3.4.15}$$

Thus $n = 15$ and xy has cycle type 3×5^2 .

(A3) The 3 points are contained in two cycles (2 points in one, 1 in the other).

$$\text{lcm}(r + s, t) = n - 2 \tag{3.4.16}$$

$$\text{lcm}(s, r + t) = n \tag{3.4.17}$$

Suppose $t = 1$. Then let $u = r + 1$, and the equations become:

$$s + u = n - 1 \quad (3.4.18)$$

$$\text{lcm}(s, u) = n \quad (3.4.19)$$

Then s and u are proper divisors of n (which is odd), yielding $s, u \leq n/3$. Thus $n - 1 = s + u \leq 2n/3$, which contradicts our assumption that $n \geq 7$. Thus $t \neq 1$ (this will be required for Lemma 3.8).

However, for $t > 1$, there are many solutions to this pair of equations. The smallest values of n for which equations (3.4.16) and (3.4.17) have a solution are given in Table 3.3, and can also be found as sequence A108157 in [47]. The smallest value is $n = 497$.

(A4) The 3 points are contained in three cycles (1 point in each).

$$\text{lcm}(r, s, t) = n - 2 \quad (3.4.20)$$

$$r + s + t = n \quad (3.4.21)$$

Suppose r, s and t are all proper divisors of $n - 2$. Then because $n - 2$ is odd, we must have:

$$\max(r, s, t) \leq (n - 2)/3 \quad (3.4.22)$$

But this is clearly incompatible with (3.4.21). So at least one of r, s and t must be equal to $n - 2$. This then forces the other two to be equal to 1, and so b is an $(n - 2)$ -cycle and ab is an n -cycle.

By the above analysis we get:

n	r	s	t	n	r	s	t
497	38	7	33	2277	142	33	65
623	62	7	27	2795	82	65	133
875	166	125	9	2873	82	17	87
1017	194	9	145	3077	106	17	75
1107	58	27	65	3215	454	5	189
1199	52	11	57	3383	130	17	69
1397	82	11	45	3629	98	19	93
1617	62	33	85	3743	110	19	87
1991	142	11	39	3885	248	105	11
2093	110	13	51	4097	178	17	63

Table 3.3: Smallest solutions to equations (3.4.16), (3.4.17)

Lemma 3.8 *Let $n \geq 7$ be odd. Suppose x' is a 3-cycle and y' is an $(n - 2)$ -cycle such that $x'y'$ has order n . Then one of the following holds:*

1. $x'y'$ is an n -cycle, and there is an automorphism of A_n mapping x' and y' to x and y .
2. (The exceptional case.) $n = 15$ and $x'y'$ is an element of cycle shape $3 \times 5 \times 5$

Definition-Theorem 3.9 *Standard generators of A_n for $n \geq 7$, n odd are given by a 3-cycle x and an $(n - 2)$ -cycle y such that xy has order n (and xy^2 has order 15 if $n = 15$).*

3.4.2 Standard generators for A_n , $n \geq 8$ even

Now we suppose n is even.

(A1) All 3 points are contained in a single cycle in the order 1, 2, 3.

We have already remarked that this is impossible.

(A2) All 3 points are contained in a single cycle in the order 1, 3, 2.

$$r + s + t = n - 1 \tag{3.4.23}$$

$$\text{lcm}(r, s, t) = n - 2 \tag{3.4.24}$$

By equation (3.4.23), r, s and t are proper divisors of $n - 2$ (even), so:

$$\max(r, s, t) \leq (n - 2)/2 \quad (3.4.25)$$

Without loss of generality, we say $r \geq s \geq t$. On the other hand, by (3.4.23), $r \geq (n - 1)/3 > (n - 2)/3$. Combining with (3.4.25) we get:

$$r = \frac{n - 2}{2} \quad (3.4.26)$$

Hence $s + t = n/2$, whence $s \geq n/4$. Thus either $s = (n - 2)/2$ or $s = (n - 2)/3$. The first possibility yields $t = 1$ by (3.4.23), contradicting (3.4.24). Thus

$$s = \frac{n - 2}{3} \quad (3.4.27)$$

Then (3.4.23) forces:

$$t = \frac{n - 2}{6} + 1 \quad (3.4.28)$$

Now t divides $n - 2$, so we can write $t = (n - 2)/r$ for $r \in \{3, 4, 5\}$. Thus:

$$\frac{n - 2}{6} + 1 = \frac{n - 2}{r} \quad (3.4.29)$$

which yields:

$$n \in \{8, 14, 32\} \quad (3.4.30)$$

(A3) The 3 points are contained in two cycles (2 points in one, 1 in the other).

$$\text{lcm}(r + s, t) = n - 1 \quad (3.4.31)$$

$$\text{lcm}(s, r + t) = n - 2 \quad (3.4.32)$$

n	r	s	t	n	r	s	t
58	11	8	3	610	55	32	21
146	13	16	5	640	49	22	9
156	17	14	5	716	31	34	11
206	29	12	5	738	35	32	11
288	19	22	7	782	41	30	11
466	139	16	93	834	55	64	49
478	25	28	9	838	49	44	27
496	29	26	9	870	51	28	11
498	55	16	7	982	89	20	9
562	107	80	33	982	187	140	9
596	31	54	35	1028	41	38	13

Table 3.4: Smallest solutions to equations (3.4.31), (3.4.32)

Firstly, suppose $t > 1$. In this case, there are a number of solutions, the smallest of which are given in Table 3.4. The relevant values of n can be found as sequence A108750 in [47]. Note however that in this situation we have:

$$[x, y] = (1, 3, 2)(1, 2, 3)^{(1, \circ, \dots, \circ, 2, \bullet, \dots, \bullet)}(3, \star, \dots, \star) \quad (3.4.33)$$

Because s is even, it is at least 2. If $r > 1$ we have:

$$[x, y] = (1, 3, 2)(\circ, \bullet, \star) \quad (3.4.34)$$

which has order 3. If $r = 1$, then we have:

$$[x, y] = (1, 3, 2)(2, \bullet, \star) = (1, 3, \bullet, \star, 2) \quad (3.4.35)$$

which has order 5. Thus in particular $o([x, y]) \neq 2$.

Otherwise suppose that $t = 1$ so that y is an $(n - 1)$ -cycle. So $r + s = n - 1$, and since $n - 1$ is odd, either $r \geq n/2$ or $s \geq n/2$. If $r \geq n/2$, then $r + 1 = n - 2$

(because $r + 1$ divides $n - 2$), and so $r = n - 3$, $s = 2$. Alternatively, if $s \geq n/2$ then $s = n - 2$ (because s divides $n - 2$), whence $r = 0$. By arbitrarily numbering the points, this gives two possibilities, which can be distinguished by looking at $o([x, y])$:

$$(a) \quad y = (1, 4, 5, \dots, n - 1, 2, n - 2), xy = (1, n - 2)(2, 3, 4, \dots, n - 1)$$

$$\begin{aligned} [x, y] &= (1, 3, 2)(1, 2, 3)^{(1, 4, 5, \dots, n-1, 2, n-2)} \\ &= (1, 3, 2)(4, n - 2, 3) \\ &= (1, 4, n - 2, 3, 2) \end{aligned} \tag{3.4.36}$$

$$(b) \quad y = (1, 2, 4, 5, \dots, n - 1), xy = (1, 4, \dots, n - 1)(2, 3)$$

$$\begin{aligned} [x, y] &= (1, 3, 2)(1, 2, 3)^{(1, 2, 4, 5, \dots, n-1)} \\ &= (1, 3, 2)(2, 4, 3) \\ &= (1, 2)(3, 4) \end{aligned} \tag{3.4.37}$$

The second case corresponds to our choice of standard generators for A_n .

(A4) The 3 points are contained in three cycles (1 point in each).

$$\text{lcm}(r, s, t) = n - 1 \tag{3.4.38}$$

$$r + s + t = n - 2 \tag{3.4.39}$$

As above, r, s and t are proper divisors of $n - 1$, which is odd. Therefore $\max(r, s, t) \leq (n - 1)/3$. Then (3.4.39) implies that the three numbers must take the values

$(n-1)/3$, $(n-1)/3$ and $(n-1)/3 - 1$ in some order. But then:

$$\text{lcm}(r, s, t) = \frac{(n-1)(n-4)}{9} = n-1 \quad (3.4.40)$$

which implies $n \in \{1, 13\}$. But n was assumed to be even, so this is a contradiction.

This analysis proves:

Lemma 3.10 *Let $n \geq 8$ be even. Suppose x' is a 3-cycle and y' is an element of order $n-1$ such that $x'y'$ has order $n-2$. Then one of the following holds:*

1. *y' is an $(n-1)$ -cycle, $x'y'$ has cycle type $2 \times (n-2)$ and $[x', y']$ has cycle type 2^2 , and there is an automorphism of A_n which maps x' and y' to x and y ;*
2. *y' is an $(n-1)$ -cycle, $x'y'$ has cycle type $2 \times (n-2)$ and $[x', y']$ is a 5-cycle;*
3. *(The exceptional cases) y' is an $(n-1)$ -cycle, $x'y'$ has cycle type $\frac{n-2}{2} \times \frac{n-2}{3} \times (\frac{n-2}{6} + 1)$ and n is one of 8, 14 or 32;*
4. *y' is the product of two non-trivial cycles and $[x', y']$ is a 3-cycle or 5-cycle.*

Example 3.11 *Let $x = (1, 2, 3)$. Here are some $(n-1)$ -cycles y_n (with n even) such that xy_n has order $n-2$ but does not have cycle shape $2 \times (n-2)$. These exhibit the exceptional cases*

of Lemma 3.10. Observe that $o([x, y_n]) = 3$ in each case.

$$y_8 = (1, 4, 5, 3, 6, 2, 7) \quad (3.4.41)$$

$$xy_8 = (1, 7)(2, 6)(3, 4, 5) \quad (3.4.42)$$

$$y_{14} = (1, 4, 5, 6, 7, 8, 3, 9, 10, 11, 2, 12, 13) \quad (3.4.43)$$

$$xy_{14} = (1, 12, 13)(2, 9, 10, 11)(3, 4, 5, 6, 7, 8) \quad (3.4.44)$$

$$y_{32} = (1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, \quad (3.4.45)$$

$$3, 18, 19, 20, 21, 22, 23, 24, 25, 26, 2, 27, 28, 29, 30, 31)$$

$$xy_{32} = (1, 27, 28, 29, 30, 31)(2, 18, 19, 20, 21, 22, 23, 24, 25, 26) \quad (3.4.46)$$

$$(3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17)$$

Definition-Theorem 3.12 *Standard generators of A_n for $n \geq 8$, n even are given by a 3-cycle x and an element y of order $(n - 1)$ such that xy has order $n - 2$ and $[x, y]$ has order 2. Note that y is necessarily an $(n - 1)$ -cycle.*

3.5 Finders for A_n

In this section, we discuss the problem of finding a finder for A_n .

If we have a 3-cycle x and a $\beta(n)$ -cycle y' , then we need to find a conjugate y of y' such that all the conditions hold.

The probability p of finding a particular conjugacy class of pair (x, y) by conjugating is given by:

$$p = \frac{|C_G(x)| \cdot |C_G(y)|}{|A_n|} \quad (3.5.1)$$

However, if n is odd, there are two conjugacy classes of pairs corresponding to standard generators, so the probability of finding a pair of standard generators is doubled. The probability p' of finding a pair of standard generators in a single conjugation is

therefore:

$$p' = \begin{cases} \frac{6}{n(n-1)} & \text{if } n \text{ is odd} \\ \frac{3}{n(n-2)} & \text{if } n \text{ is even} \end{cases} \quad (3.5.2)$$

So the problem is reduced to finding a 3-cycle and a $\beta(n)$ -cycle.

3.5.1 Finding a 3-cycle

Let \mathcal{R}_n denote the class of 3-cycles in A_n . We can find a 3-cycle in a way similar to how we found transpositions in S_n ; find integers m such that all elements in A_m of order m power up to 3-cycles. A list of taming sets $T(\mathcal{R}_n)$ for small n is given in Table 3.5. Analogously to Theorem 3.6 we have:

Theorem 3.13 *Let $n \geq 5$, $n \neq 6$ be an integer. Then:*

1. *There exists an integer r_n such that all elements of order r_n in A_n power up to give a 3-cycle.*
2. *The taming set $T(\mathcal{R}_n)$ is non-empty.*

Proof. For $n \leq 30$, consult Table 3.5. For $n > 30$, we can write:

$$n - 3 = \sum_{i=1}^k p_i \quad (3.5.3)$$

where the p_i are distinct primes greater than or equal to 5 (this is by a result of Kløve [30] analogous to Lemma 3.5). Then set $r_n = 3 \prod p_i$ as before. Any element of S_n with this order must have $k + 1$ cycles of orders $3, p_1, \dots, p_k$ respectively, and hence must be in A_n and must power up to a 3-cycle. ■

3.5.2 Finding a $\beta(n)$ -cycle

For $n \leq 25$, $n \neq 16, 17, 22, 23$, all elements of order $\beta(n)$ in A_n are $\beta(n)$ -cycles. The probability of finding an $\beta(n)$ -cycle by a random search in A_n is $2/\beta(n)$.

n	$T(\mathcal{R}_n)$	$p \approx$
5	3	1/3
6	3^a	1/4
7	6	1/12
8	15	1/7
9	12, 15	1/5
10	15, 21	1/6
11	21	1/10
12	21, 30	1/16
13	24	1/24
14	33, 42, 60	1/11
15	33, 105	1/13
16	33, 39, 84, 105	1/9
17	39, 105	1/16
18	39, 66, 120	1/24
19	165, 210	1/69
20	51, 78, 132, 165, 168	1/14
21	48, 51, 165, 195, 231, 420	1/11
22	51, 57, 156, 195, 231	1/12
23	57, 195, 231, 273, 330	1/19
24	57, 102, 264, 273	1/30
25	255, 273, 390, 462, 660, 840	1/60
26	69, 114, 204, 240, 255, 312, 1155	1/18
27	69, 255, 285, 357, 429, 546, 780, 924, 1155	1/18
28	69, 75, 228, 285, 336, 357, 429, 1155, 1365	1/15
29	75, 285, 357, 399, 429, 510, 1092, 1320, 1365	1/23
30	75, 138, 399, 408, 1365, 2310	1/39

^aAll 3-elements in A_6 fuse into one class in $\text{Aut}(A_6)$

Table 3.5: Taming sets for the class \mathcal{R}_n of 3-cycles in A_n

n	Γ_n^+	Γ_n^-
16	8, 14, 22, 28, 36, 40, 42, 63	3, 4, 10, 11, 21, 35
17	7, 8, 14, 17, 22, 28, 36, 40, 42, 63	3, 4, 10, 11, 21
22	10, 16, 18, 19, 20, 30, 34, 36, 40, 48, 52, 55, 57, 70, 72, 78, 85, 88, 90, 99, 117, 120, 132, 140, 195, 315	3, 5, 6, 8, 13, 14, 17, 28, 33, 35, 39, 45, 63, 77
23	10, 16, 18, 19, 20, 23, 30, 34, 36, 40, 48, 52, 55, 57, 70, 72, 78, 85, 88, 90, 99, 117, 120, 132, 140, 195, 315	3, 5, 6, 8, 13, 14, 17, 28, 33, 35, 39, 45, 63, 77, 273

Table 3.6: Sets Γ_n^\pm for distinguishing $\beta(n)$ -cycles

For $n \in \{16, 17, 22, 23\}$, we can find sets Γ_n^+ and Γ_n^- satisfying the following property:

Property 3.14 *Let t be an element of A_n with order $\beta(n)$. Then:*

1. *If $o(xt) \in \Gamma_n^+$ then t is an $\beta(n)$ -cycle.*
2. *If $o(xt) \in \Gamma_n^-$ then t is not an $\beta(n)$ -cycle.*
3. *For some conjugate t' of t , $o(xt') \in \Gamma_n^+ \cup \Gamma_n^-$.*

Given such sets, we can identify an $\beta(n)$ -cycle by taking a random element t of order $\beta(n)$ with probability approximately $2/\beta(n)$, and choose conjugates t' of t until the order of xt' lies in either Γ_n^+ or Γ_n^- . If the order is in Γ_n^+ , then we have found an $\beta(n)$ -cycle, and if not, then we have found an element which is definitely not an $\beta(n)$ -cycle, so we need to return to the beginning. Sets fulfilling Property 3.14 are given in Table 3.6.

Alternatively for $n = 17, 23$, since n is prime, all elements of order n are n -cycles. Find an element t of order n (probability $2/n$) and then find a conjugate t' of t such that xt' has order $n - 2$ (probability $3/((n - 2)(n - 1))$). Then xt' is the required $(n - 2)$ -cycle. In fact, because of the remark following Lemma 3.8, x^{-1} and xt' are standard

generators of A_n .

Alternatively for $n = 22$, find an element t of order 19 ($p = 1/57$) and find a conjugate t' of it such that xt' has order 21 ($p \approx 1/54$). Then xt' is an $(n - 1)$ -cycle.

3.6 Checkers for A_n

To check elements x, y are standard generators of A_n , we need some way of showing that x is a 3-cycle and y is an $\beta(n)$ -cycle. If we want a general scheme, we can use the following presentations due to Carmichael [9, Theorem 22]:

Theorem 3.15 *Let $n > 3$. For n odd we have:*

$$A_n = \langle s, t \mid s^{n-2} = t^3 = (st)^n = (ts^{-k}ts^k)^2 = 1 \ (1 \leq k \leq (n-3)/2) \rangle \quad (3.6.1)$$

The permutations $\sigma = (3, \dots n), \tau = (1, 2, 3)$ satisfy this presentation.

For n even we have:

$$A_n = \langle s, t \mid s^{n-2} = t^3 = (st)^{n-1} = (ts^{(-1)^k}s^{-k}ts^k)^2 = 1 \ (1 \leq k \leq (n-2)/2) \rangle \quad (3.6.2)$$

The permutations $\sigma = (1, 2)(3, \dots n), \tau = (1, 2, 3)$ satisfy this presentation.

To use these presentations, we set $t = x, s = xy$ (where x and y are our standard generators).

Alternatively, we can provide checkers for $7 \leq n \leq 30$ as follows:

3.6.1 Checking x is a 3-cycle

The first task is to show that x is a 3-cycle.

Lemma 3.16 *If $x, y \in G$ satisfy $o(x) = o(y) = 3$ and $o(xy) = 2$ then x is G -conjugate to y^{-1} . In particular, if x and y are permutations then they have the same cycle shape.*

Proof. Let $H = \langle x, y \rangle$. Then because

$$A_4 = \langle x, y \mid x^3 = y^3 = (xy)^2 = 1 \rangle \quad (3.6.3)$$

we must have $H \cong A_4$. The elements representing x and y in A_4 can be taken to be $(1, 2, 3)$ and $(2, 3, 4)$, so x is H -conjugate to y^{-1} . ■

Hence we can prove that x is a 3-cycle by finding an element t whose order o is in $T(\mathcal{R}_n)$, and then find an element u such that $o(x(t^{o/3})^u) = 2$. Words giving such t and u are given in Table 3.7.

3.6.2 Checking y is an $\beta(n)$ -cycle

Suppose x is known to be a 3-cycle, and we wish to show that y is an $\beta(n)$ -cycle. By the analysis in section 3.4.1, if n is odd, the standard relations are enough to show that y is a $\beta(n)$ -cycle providing $n < 497$. By Definition-Theorem 3.12, for n even the standard relations are enough to show y is a $\beta(n)$ -cycle. Hence for $7 \leq n \leq 30$, we have the following semi-presentation for A_n :

$$\begin{aligned} A_n \approx \langle \langle x, y, (t, u) \mid \mathbf{std}, o(x(t^{o/3})^u) = 2, o(t) = o; \\ t, u = \text{words in } x \text{ and } y \rangle \rangle \end{aligned} \quad (3.6.4)$$

where t, u and o are as given in Table 3.7, and the standard relations ‘**std**’ are as given in Definition-Theorems 3.9 and 3.12.

n	t	$o \in T(\mathcal{R}_n)$	u
7	xy^2xyxy	6	x^2y
8	xy^2	15	x^2y^2
9	x^2y^2xy	15	xy^6
10	xy^3	21	xy^2xy^3
11	x^2y^2xy	21	$xyxy^3$
12	xy^3xy^2	30	xy^5
13	x^2y^4xyxy	24	x^2y^{10}
14	xy^4	33	x^2y^4
15	x^2y^5xy	33	xy^{12}
16	xy^8	39	1
17	x^2y^6xy	39	xy^7
18	xy^6xy^2	66	xy^8
19	xy^5xy^2xy	165	xy^8
20	xy^6	51	x^2y^6
21	x^2y^8xy	51	xy^{18}
22	xy^{11}	57	1
23	x^2y^9xy	57	xy^{10}
24	$xy^{19}xy^3$	57	xy^{20}
25	xy^8xy^2xy	255	xy^{11}
26	xy^8	69	x^2y^8
27	$x^2y^{11}xy$	69	xy^{24}
28	xy^{14}	75	1
29	$x^2y^{12}xy$	75	xy^{13}
30	$xy^{12}xy^2$	138	xy^{14}

Table 3.7: Words t, u for checking that x is a 3-cycle in (3.6.4)

Chapter 4

The linear groups $L_2(q)$

In this chapter, we will define standard generators for the groups in \mathcal{L}^G of the form $L_2(q)$.

4.1 Basic facts about $L_2(q)$

Let $G = \text{PSL}_2(q) = L_2(q)$ for $q = p^r$ a prime power. We will require the following facts about G in the sequel:

Lemma 4.1 (Dickson [24]) *Any maximal subgroup of G must be one of the following:*

- *a Borel subgroup: the semi-direct product of C_p^r by a cyclic group of order $(q - 1)/2$ (or $q - 1$ if q is even)*
- *a dihedral group*
- *a subfield group: $L_2(p^m)$ or $L_2(p^m).2$ where m divides r .*
- *one of S_5 , A_5 and A_4*

(Dickson gives a more precise statement of this lemma, but this weaker version is sufficient for our purposes.)

χ	Parameter	$\chi(1)$	$\chi(z)/\chi(1)$
$\mathbf{1}$		1	1
ψ		q	1
χ_i	$1 \leq i \leq (q-3)/2$	$q+1$	$(-1)^i$
θ_j	$1 \leq j \leq (q-1)/2$	$q-1$	$(-1)^j$
ξ_k	$k \in \{1, 2\}$	$(q+1)/2$	$(-1)^{(q-1)/2}$
η_k	$k \in \{1, 2\}$	$(q-1)/2$	$(-1)^{(q+1)/2}$

Table 4.1: Characters of $\mathrm{SL}_2(q)$, q odd (z is the central element of order 2)

Lemma 4.2 *Let $G = \mathrm{L}_2(p^r)$. The group G has a unique conjugacy class $2A$ of involutions, with size given by:*

$$|2A| = \begin{cases} q^2 - 1 & \text{if } q \equiv 0 \pmod{2} \\ q(q-1)/2 & \text{if } q \equiv -1 \pmod{4} \\ q(q+1)/2 & \text{if } q \equiv 1 \pmod{4} \end{cases} \quad (4.1.1)$$

Lemma 4.3 *Let $G = \mathrm{L}_2(p^r)$. If $p \neq 3$ then G has a unique conjugacy class $3A$ of elements of order 3, with size given by:*

$$|3A| = \begin{cases} q(q-1) & \text{if } q \equiv -1 \pmod{3} \\ q(q+1) & \text{if } q \equiv 1 \pmod{3} \end{cases} \quad (4.1.2)$$

If $p = 3$ then G has 2 conjugacy classes $3A$ and $3B$ of elements of order 3, with sizes given by:

$$|3A| = |3B| = (q^2 - 1)/2 \quad (4.1.3)$$

These classes are swapped by an outer automorphism of G .

Lemma 4.4 *Let q be an odd prime power. Then $\mathrm{SL}_2(q)$ has $q+4$ irreducible characters, parameterised as in Table 4.1.*

χ	Parameter	$\chi(1)$
$\mathbf{1}$		1
ψ		q
χ_i	$1 \leq i \leq (q-2)/2$	$q+1$
θ_j	$1 \leq j \leq q/2$	$q-1$

Table 4.2: Characters of $\mathrm{SL}_2(q)$, q even

Lemma 4.5 *Let q be a power of 2. Then $\mathrm{L}_2(q) \cong \mathrm{SL}_2(q)$ has $q+1$ irreducible characters, parameterised as in Table 4.2.*

4.2 Standard generators for $\mathrm{L}_2(p)$

In order to define standard generators for $G = \mathrm{L}_2(q)$, $q = p^r$, we will consider three cases:

- $r = 1$ (*prime case*)
- $r > 1$ and $p = 2$ (*even prime-power case*)
- $r > 1$ and $p > 2$ (*odd prime-power case*)

Define:

$$\lambda(q) = \begin{cases} q & \text{prime case} \\ q-1 & \text{even prime-power case} \\ (q-1)/2 & \text{odd prime-power case} \end{cases} \quad (4.2.1)$$

Then we get the following important consequence of Lemma 4.1.

Theorem 4.6 *Let $q \geq 7$ and let (x, y) be a $(2, 3, \lambda(q))$ -pair in G . Then G is generated by x and y .*

Proof. Suppose not. Then let $H = \langle x, y \rangle$, and so $H \leq M$ for some maximal subgroup M of G . Clearly M is not dihedral, and M cannot be S_5 , A_5 or A_4 . Moreover, for $m < r$, no

element of $L_2(p^m)$ can have order $\lambda(q)$ (the highest element order in $L_2(p^m)$ is $p^m + 1$ or $(p^m + 1)/2$) so M cannot be a subfield group. So M must be a Borel subgroup. Consider \overline{M} , the image of the Borel subgroup modulo the elementary abelian group of order p^r . Then \overline{M} is cyclic. For $g \in M$, we either have $o(\overline{g}) = o(g)$ or $o(\overline{g}) = o(g)/p$. Because \overline{M} is cyclic, we also have:

$$o(\overline{xy}) = o(\overline{x}\cdot\overline{y}) = \text{lcm}(o(\overline{x}), o(\overline{y})) \quad (4.2.2)$$

But this is incompatible with (x, y) being a $(2, 3, \lambda(q))$ -pair. ■

Definition 4.7 A semi-standard pair for $L_2(q)$ is a pair (x, y) such that $o(x) = 2$, $o(y) = 3$ and $o(xy) = \lambda(q)$.

We do not yet know that semi-standard pairs exist, but by Theorem 4.6, all semi-standard pairs generate G . In the following sections we complete the definition of standard generators for $L_2(q)$ for $q \leq 499$.

4.2.1 Standard generators for $L_2(p)$ (prime case)

Let $G = L_2(p)$ for a prime $p \geq 7$. A semi-standard pair for G is a $(2, 3, p)$ -pair.

Lemma 4.8 Let $p \geq 5$ be a prime. Then $L_2(p)$ contains two conjugacy classes $\mathcal{C}_p^A, \mathcal{C}_p^B$ of elements of order p . The two classes are automorphic, and their sizes are given by

$$|\mathcal{C}_p^A| = |\mathcal{C}_p^B| = (p-1)(p+1)/2 \quad (4.2.3)$$

By Theorem 4.6, all $(2, 3, p)$ -pairs generate G . We will show:

$$\zeta(2A, 3A, \mathcal{C}_p^\epsilon) = 1 \quad (4.2.4)$$

for $\epsilon \in \{A, B\}$. This implies that $(2, 3, p)$ -pairs exist and are determined up to automorphisms of $L_2(p)$, and thus no extra conditions are required to get a definition of standard generators. By section 1.4, this structure constant is the product of the quantities α and β , where:

$$\alpha = \frac{|2A||3A||\mathcal{C}_p^\epsilon|}{|L_2(p)|^2} \quad (4.2.5)$$

and

$$\beta = \sum_{\chi \in L_2(p)} t_\chi \quad (4.2.6)$$

where

$$t_\chi = \frac{\chi(2A)\chi(3A)\overline{\chi(\mathcal{C}_p^\epsilon)}}{\chi(1)} \quad (4.2.7)$$

The proof that $\alpha\beta = 1$ seems to require a case-by-case analysis depending on the residue of p modulo 12.

By Lemma 4.8 and the fact that $|L_2(p)| = p(p-1)(p+1)/2$, we can easily show that:

$$\alpha = \begin{cases} (p+1)/(p-1) & \text{if } p \equiv 1 \pmod{12} \\ (p-1)/(p+1) & \text{if } p \equiv -1 \pmod{12} \\ 1 & \text{if } p \equiv \pm 5 \pmod{12} \end{cases} \quad (4.2.8)$$

In order to calculate β , we need a character table for $L_2(p)$. This can be read from the character table of $SL_2(p)$, which was known by Schur [45], and is given in [17, section 38]. The relevant character entries are given in Table 4.3, which uses the following notational device:

Notation 4.9 *We adopt the following notation for periodic functions on \mathbb{Z} :*

$$(a_1, a_2, \dots, a_n)_i = a_j, \text{ where } j \text{ is the residue of } i \text{ modulo } n \quad (4.2.9)$$

χ	$\chi(1)$	$\chi(g_2)$		$\chi(g_3)$		$\chi(g_p)$
		$p \pmod{4}$		$p \pmod{3}$		
		1	-1	1	-1	
1	1	1	1	1	1	1
ψ	p	1	-1	1	-1	0
χ_{2i}	$p+1$	$2(-1)^i$	0	$(-1, -1, 2)_i$	0	1
θ_{2j}	$p-1$	0	$2(-1)^{j+1}$	0	$(1, 1, -2)_i$	-1
ξ_k	$(p+1)/2$	$(-1)^{(p-1)/4}$	—	1	0	$\frac{1}{2}(-1 \pm \sqrt{p})$
η_k	$(p-1)/2$	—	$(-1)^{(p-3)/4}$	0	-1	$\frac{1}{2}(-1 \pm \sqrt{-p})$

Table 4.3: Some character values of $L_2(p)$, $p \geq 5$ odd

For example, we have that $\omega^i + \omega^{2i} = (-1, -1, 2)_i$ for all $i \in \mathbb{Z}$ (where $\omega = \exp(2\pi i/3)$).

To calculate β , we will need the values of each irreducible character on the class representatives. This information is given in Table 4.3.

We will need to split into 4 cases, depending on the residue of p modulo 12. For convenience we define for each $r \geq 0$:

$$\Delta_r = \sum_{j=1}^r (-1)^j (-1, -1, 2)_j \quad (4.2.10)$$

Lemma 4.10 *We have $\Delta_r = (1, 0, -2, -3, -2, 0)_r$.*

Proof. Direct calculation gives the first 6 values. Since $\Delta_6 = 0$ and the i th summand of equation (4.2.10) only depends on $i \pmod{6}$, the result follows. ■

Case $p \equiv 1 \pmod{12}$

We have $p \equiv 1 \pmod{3}$ and $p \equiv 1 \pmod{4}$. Then:

$$\begin{aligned}\beta &= t_1 + \sum_{j=1}^{(p-5)/4} t_{\chi_{2j}} + t_{\xi_1} + t_{\xi_2} \\ &= 1 + \frac{2\Delta_{(p-5)/4}}{p+1} + \frac{2(-1)^{(p-1)/4}}{p+1} \\ &= \frac{p-1}{p+1}\end{aligned}\tag{4.2.11}$$

because $(p-5)/4$ must be congruent to either 2 or 5 modulo 6.

Case $p \equiv -1 \pmod{12}$

This case is similar to the previous one. We have $p \equiv -1 \pmod{3}$ and $p \equiv -1 \pmod{4}$. Then

$$\begin{aligned}\beta &= t_1 + \sum_{j=1}^{(p-3)/4} t_{\theta_{2j}} + t_{\eta_1} + t_{\eta_2} \\ &= 1 - \frac{2\Delta_{(p-3)/4}}{p-1} - \frac{2(-1)^{(p+1)/4}}{p-1} \\ &= \frac{p+1}{p-1}\end{aligned}\tag{4.2.12}$$

because $(p-3)/4$ must be congruent to either 2 or 5 modulo 6.

Case $p \equiv 5 \pmod{12}$

We have $p \equiv -1 \pmod{3}$ and $p \equiv 1 \pmod{4}$, and so the only term in the sum for which $t_\chi \neq 0$ is when $\chi = \mathbf{1}$. Thus $\beta = 1$.

Case $p \equiv -5 \pmod{12}$

This case is similar to the previous one. We have $p \equiv 1 \pmod{3}$ and $p \equiv -1 \pmod{4}$, and so the only χ for which $t_\chi \neq 0$ is when $\chi = \mathbf{1}$. Thus $\beta = 1$.

Hence we have proved:

Definition-Theorem 4.11 *Standard generators of $L_2(p)$ are x and y such that x has order 2, y has order 3 and xy has order p .*

We observe that (for reasonably small p) it is easy to find elements of the correct orders, and by conjugating by elements, it is easy to find a pair of standard generators. For large primes p , better methods would be needed. Since we are only concerned with complex representations of dimension less than or equal to 250, the largest value of p that we need to consider is $p = 499$ (since 501 is not prime).

Remark The calculation of $\zeta(2A, 3A, \mathcal{C}_p^\epsilon)$ can be automated by using the CHEVIE [20] computer package. Among other things, this package can calculate structure constants from *generic character tables*. These are character tables for a family of Chevalley groups with a fixed Dynkin diagram: the entries are a function of the field size q . In the case $L_2(q)$, for instance, there are three generic character tables to consider, depending on the residue of q modulo 4.

4.2.2 Standard generators for $L_2(2^r)$ (even prime-power case)

Let $G = L_2(q)$ for $q = 2^r$, $r \geq 3$. A semi-standard pair for G is a $(2, 3, q - 1)$ -pair.

These conditions are usually not enough to determine the pair up to automorphisms of the group.

Lemma 4.12 *The group $G = L_2(q)$ ($q = 2^r$) contains $\phi(q - 1)/2$ classes \mathcal{C}_{q-1}^i ($1 \leq i \leq \phi(q - 1)/2$) of elements of order $q - 1$, arranged in $\kappa(r) = \phi(q - 1)/(2r)$ sets of r which fuse in $\text{Aut}(L_2(q))$. Each class has size:*

$$|\mathcal{C}_{q-1}^i| = q(q + 1) \quad (4.2.13)$$

Theorem 4.13 For all classes \mathcal{C}_{q-1}^i of elements of order $q - 1$, we have:

$$\zeta(2A, 3A, \mathcal{C}_{q-1}^i) = 1 \quad (4.2.14)$$

In particular, semi-standard pairs exist.

Proof. As before we have $\zeta(2A, 3A, \mathcal{C}_{q-1}^i) = \alpha\beta$, with:

$$\alpha = \frac{|2A||3A||\mathcal{C}_p^i|}{|\mathbf{L}_2(q)|^2} \quad (4.2.15)$$

and:

$$\beta = \sum_{\chi \in \text{Irr}(\mathbf{L}_2(q))} t_\chi \quad (4.2.16)$$

where

$$t_\chi = \frac{\chi(2A)\chi(3A)\overline{\chi(\mathcal{C}_{q-1}^i)}}{\chi(1)} \quad (4.2.17)$$

Using Lemma 4.12 and the fact that $|\mathbf{L}_2(q)| = q(q^2 - 1)$ we have:

$$\begin{aligned} \alpha &= \frac{(q^2 - 1)q(q \pm 1)q(q + 1)}{q^2(q^2 - 1)^2} \\ &= \begin{cases} (q + 1)/(q - 1) & \text{if } r \text{ is even} \\ 1 & \text{if } r \text{ is odd} \end{cases} \end{aligned} \quad (4.2.18)$$

If r is odd, then $q \equiv -1 \pmod{3}$, and the only t_χ which is non-zero is when $\chi = 1$. Hence $\beta = 1$, giving $\zeta = 1$.

Otherwise, r is even, and we have:

$$\begin{aligned}\beta &= t_1 + \sum_{j=1}^{(q-2)/2} t_{\chi_j} \\ &= 1 + \sum_{j=1}^{(q-2)/2} \frac{(\omega^j + \omega^{-j})(\rho^j + \rho^{-j})}{q+1}\end{aligned}\tag{4.2.19}$$

where ρ is a primitive $(q-1)$ th root of unity. Using the fact that $(\omega^j + \omega^{-j}) = (-1, -1, 2)_j$, we can rewrite this as:

$$\begin{aligned}\beta &= 1 + \frac{1}{q+1} \left(- \sum_{j=1}^{(q-2)/2} (\rho^j + \rho^{-j}) + 3 \sum_{j=1, 3|j}^{(q-2)/2} (\rho^j + \rho^{-j}) \right) \\ &= 1 + \frac{1}{q+1} \left(1 + 3 \sum_{k=1}^{(q-4)/6} (\rho^{3k} + \rho^{-3k}) \right)\end{aligned}\tag{4.2.20}$$

Since $q-1$ is divisible by 3, ρ^3 is a primitive $((q-1)/3)$ th root of unity, so:

$$\beta = 1 - \frac{2}{q+1} = \frac{q-1}{q+1}\tag{4.2.21}$$

Thus $\zeta = 1$. (Again, we could have used CHEVIE to prove this result.) ■

Conditions that can be used to single out a semi-standard pair for $r \leq 14$ are given in Table 4.4. They were found by finding $\kappa(r) = \phi(q-1)/(2r)$ semi-standard pairs which all have different fingerprints, and selecting a condition which eliminated all but one of the pairs. Since all of the classes of pairs are equally likely to be found in a random search, we selected conditions involving smaller orders (to make calculations faster).

Although the table goes further, the largest value of q that we will need to consider is $q = 128$, since the smallest (faithful) complex representation of $L_2(256)$ has dimension 255. For very large r , this choice of standard generators is not ideal, because only

r	q	$\kappa(r)$	Extra conditions
3	8	1	—
4	16	1	—
5	32	3	$o(xyxy^2) = 11$
6	64	3	$o(xyxy^2) = 13$
7	128	9	$o(xyxyxy^2) = 43$
8	256	8	$o(xyxy^2) = 85$
9	512	24	$o(xyxy^2) = 27$
10	1024	30	$o(xyxy^2) = 41$
11	2048	88	$o(xyxy^2) = 23$
12	4096	72	$o(xyxy^2) = 91$
13	8192	315	$o(xyxy^2) = o(xyxyxy^2) = o(xyxyxy^2xy^2) = 2731$
14	16384	378	$o(xyxyxy^2) = 43$

Table 4.4: Extra conditions for $L_2(q)$ (even prime-power case)

Group	$o(x)$	$o(y)$	$o(xy)$	Other conditions and notes
$L_2(9)$	2	4	5	$G \cong A_6$
$L_2(25)$	2	3	13	$o(xyxyy) = 4$ (forces $xy \in 13A/B$)
$L_2(27)$	2	3	7	
$L_2(49)$	2	3	25	$o(xyxyy) = 24$

Table 4.5: Odd prime-power standard generators from the Web Atlas

$1/2^r$ of the elements of $L_2(2^r)$ have even order.

4.2.3 Standard generators for $L_2(p^r)$ (odd prime-power case)

Finally, we must consider fields $GF(q)$ where q is a power of a prime $p > 2$. The Web ATLAS already contains definitions for $q \in \{9, 25, 27, 49\}$, and these are given in Table 4.5. The field sizes q where $L_2(q)$ has a representation of size ≤ 250 that we have not yet considered are therefore 81, 121, 125, 169, 243, 289, 343 and 361.

Recall that a semi-standard pair for G is a pair (x, y) such that $o(x) = 2$, $o(y) = 3$ and $o(xy) = (q - 1)/2$. By Theorem 4.6 each semi-standard pair generates G .

Lemma 4.14 *Let $G = L_2(q)$, $q = p^r$, $r > 1$, $p > 2$. Then:*

1. G contains $\phi((q - 1)/2)/2$ conjugacy classes of elements of order $(q - 1)/2$.

2. If $p = 3$ and $\mathcal{C}_{(q-1)/2}$ is any one of these conjugacy classes then:

$$\xi_G(2A, 3A, \mathcal{C}_{(q-1)/2}) = 1 \quad (4.2.22)$$

$$\xi_G(2A, 3B, \mathcal{C}_{(q-1)/2}) = 1 \quad (4.2.23)$$

3. If $p > 3$ and $\mathcal{C}_{(q-1)/2}$ is any one of these conjugacy classes then:

$$\xi_G(2A, 3A, \mathcal{C}_{(q-1)/2}) = 2 \quad (4.2.24)$$

4. $\text{Out}(G) \cong C_2 \times C_r$

Proof. Fact 1 follows from the character table of G , and facts 2 and 3 can be proved using CHEVIE. Fact 4 is well-known. ■

Theorem 4.15 *There are $\frac{\phi((q-1)/2)}{2r}$ automorphism classes of semi-standard pairs in G .*

Proof. By Lemma 4.14, the number of conjugacy classes of semi-standard pairs in G is $\phi((q-1)/2)$. This means there are $\phi((q-1)/2)/(2r)$ automorphism classes. ■

As for the even prime-power case, we searched at random for enough semi-standard pairs with distinct fingerprints to ensure that we had found representatives of all the automorphism classes of semi-standard pairs. We then selected a single condition which uniquely specifies one of the classes. The conditions we chose for standard generators are given in Table 4.6.

4.3 Black box algorithms for $L_2(q)$

By Lemmas 4.2 and 4.3, the classes $2A$ and $3A$ (or $3A/B$ if q is divisible by 3) are completely tame. Therefore we can use the standard finder (Algorithm 1.11) to find generators for $L_2(q)$.

G	Extra condition
$L_2(81)$	$o(xyxyxy^2) = 40$
$L_2(121)$	$o(xyxy^2) = 12$
$L_2(125)$	$o(xyxy^2) = 21$
$L_2(169)$	$o(xyxy^2) = 14$
$L_2(243)$	$o(xyxyxy^2) = 11$
$L_2(289)$	$o(xyxy^2) = 16$
$L_2(343)$	$o(xyxy^2) = 9$
$L_2(361)$	$o(xyxy^2) = 15$
$L_2(529)$	$o(xyxy^2) = 8$
$L_2(625)$	$o(xyxy^2) = 26$
$L_2(729)$	$o(xyxy^2) = 26$
$L_2(841)$	$o(xyxy^2) = 60$
$L_2(961)$	$o(xyxy^2) = 13$

Table 4.6: Extra conditions for $L_2(q)$ (odd prime-power case)

The situation for checkers is just as straightforward. Because there are unique conjugacy classes of elements of orders 2 and 3 (up to automorphisms of G), the standard relations are enough:

$$L_2(q) \approx \langle \langle x, y \mid \mathbf{std} \rangle \rangle. \quad (4.3.1)$$

Chapter 5

The symplectic groups $S_4(q)$

In this chapter, we will define standard generators for the symplectic group $S_4(q)$ where the characteristic p is at least 3. We recall that we are using ATLAS notation, so that $S_4(q) = \text{PSp}_4(q)$ is a simple group, while $\text{Sp}_4(q)$ denotes the subgroup of $\text{GL}_4(q)$ preserving a non-degenerate symplectic form.

5.1 Standard generators for $S_4(q)$, $q = p^r$, $p > 3$ prime

5.1.1 Semi-standard pairs

Let $p > 3$ be a prime, $q = p^r$, $G = S_4(q)$. We will show how to $(2, 3)$ -generate the group G .

By the information in Srinivasan's paper, G contains two classes $2A$, $2B$ of involutions and two classes $3A$, $3B$ of elements of order 3. If we look at the 4-dimensional (projective) representation of $S_4(q)$ over $\text{GF}(q)$ and diagonalise representatives of these classes over an algebraic extension of $\text{GF}(p)$, we get the following matrices:

$$2A : \begin{pmatrix} 1 & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & -1 & \cdot \\ \cdot & \cdot & \cdot & -1 \end{pmatrix}, \quad 2B : \begin{pmatrix} i & \cdot & \cdot & \cdot \\ \cdot & -i & \cdot & \cdot \\ \cdot & \cdot & i & \cdot \\ \cdot & \cdot & \cdot & -i \end{pmatrix} \quad (5.1.1)$$

$$3A : \begin{pmatrix} \omega & \cdot & \cdot & \cdot \\ \cdot & \omega^2 & \cdot & \cdot \\ \cdot & \cdot & \omega & \cdot \\ \cdot & \cdot & \cdot & \omega^2 \end{pmatrix}, \quad 3B : \begin{pmatrix} \omega & \cdot & \cdot & \cdot \\ \cdot & \omega^2 & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot \\ \cdot & \cdot & \cdot & 1 \end{pmatrix} \quad (5.1.2)$$

(where ω is a cube root of unity, and i is a square root of -1).

The group $S_4(q)$ is isomorphic to $O_5(q)$; this is shown by taking a 5-dimensional submodule of the exterior square of the natural 4-dimensional $\text{GF}(q)$ -module for $\text{Sp}_4(q)$. This gives these Jordan forms for the classes $2A/B, 3A/B$ in $O_5(q)$:

$$2A : \begin{pmatrix} -1 & \cdot & \cdot & \cdot & \cdot \\ \cdot & -1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & -1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & -1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 \end{pmatrix}, \quad 2B : \begin{pmatrix} -1 & \cdot & \cdot & \cdot & \cdot \\ \cdot & -1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 \end{pmatrix} \quad (5.1.3)$$

$$3A : \begin{pmatrix} \omega & \cdot & \cdot & \cdot & \cdot \\ \cdot & \omega^2 & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 \end{pmatrix}, \quad 3B : \begin{pmatrix} \omega & \cdot & \cdot & \cdot & \cdot \\ \cdot & \omega^2 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \omega & \cdot & \cdot \\ \cdot & \cdot & \cdot & \omega^2 & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 \end{pmatrix} \quad (5.1.4)$$

The matrices for $O_5(q)$ show that to generate G , we must take elements from classes $2B$ and $3B$, since any other combination would leave invariant a 1-dimensional subspace of the 5-dimensional module.

Definition 5.1 Let $p > 3$ be prime, $q = p^r$. A semi-standard pair for $S_4(q)$ is a pair (x, y) of elements of $S_4(q)$ such that x is a $2B$ element, y is a $3B$ element and xy has order $(q^2 + 1)/2$.

In order to show that each semi-standard pair is a generating pair for $S_4(q)$, we will need the following definition and lemma.

Definition 5.2 We define (following Berczky [3]):

- A Singer subgroup of $\text{GL}_n(q)$ is a cyclic subgroup of order $q^n - 1$.
- Let $G \leq \text{GL}_n(q)$ be a finite classical group. A Singer subgroup of G is an irreducible cyclic subgroup of G of maximal possible order.

A generator of a Singer subgroup is called a Singer element.

Singer subgroups of classical groups need not exist, although when they exist, they are the intersections of Singer subgroups of $\mathrm{GL}_n(q)$ with G . Note that this is not a pure group theoretic definition because an abstract group can be a classical group in two different ways. Symplectic groups always have Singer elements, and they have order $(q^2 + 1)/2$.

Lemma 5.3 *For $q \neq 3$, a Singer subgroup in $\mathrm{Sp}_4(q)$ has a unique maximal overgroup: it is a subgroup of $\mathrm{Sp}_4(q)$ of field extension type (type C_3 in Aschbacher's classification).*

Proof. See the main theorem in Berczky [3]. ■

Theorem 5.4 *Each semi-standard pair for $S_4(q)$ generates $S_4(q)$.*

Proof. Suppose not; then there exists a proper subgroup H of $S_4(q)$ containing elements x and y (with x in class $2B$, y in class $3B$ and xy with order $(q^2 + 1)/2$). We will transfer our work into the double cover $\tilde{G} = \mathrm{Sp}_4(q)$, where the semi-standard pair (x, y) lifts to a pair (\tilde{x}, \tilde{y}) . Then \tilde{x} , \tilde{y} and -1 are contained in a proper subgroup $\tilde{H} < \mathrm{Sp}_4(q)$. Because either $\tilde{x}\tilde{y}$ or $-\tilde{x}\tilde{y}$ has order $q^2 + 1$, \tilde{H} contains a Singer subgroup. By Lemma 5.3, \tilde{H} is a subgroup of $\tilde{L} \cong \mathrm{Sp}_2(q^2) : 2 = \mathrm{SL}_2(q^2) : 2$. This group contains a unique conjugacy class of elements of order 3. Treating the original 4-dimensional $\mathrm{GF}(q)\tilde{G}$ -module as a 2-dimensional $\mathrm{GF}(q^2)\tilde{L}$ -module, the element \tilde{y} (like all $3B$ elements) fixes a 1-dimensional subspace: see (5.1.2). But the elements of order 3 in $\mathrm{SL}_2(q^2) : 2$ have no fixed points in the natural action. This yields a contradiction. ■

The following lemma shows that to determine a semi-standard pair (x, y) up to automorphism, it is sufficient to specify the conjugacy class for the product xy .

Lemma 5.5 *Let \mathcal{C} be any conjugacy class of G containing elements of order $(q^2 + 1)/2$.*

x^G	1	2B		3B		\mathcal{C}
$q \equiv$		+1 (mod 4)	-1 (mod 4)	+1 (mod 3)	-1 (mod 3)	
$ C_G(x) $	$\frac{1}{2}q^4(q^2 - 1)(q^4 - 1)$	$q(q - 1)(q^2 - 1)$	$q(q + 1)(q^2 - 1)$	$\frac{1}{2}q(q - 1)(q^2 - 1)$	$\frac{1}{2}q(q + 1)(q^2 - 1)$	$\frac{1}{2}(q^2 + 1)$
1	1	1	1	1	1	1
θ_9	$\frac{1}{2}q(1 + q)^2$	$1 + q$	0	$1 + q$	0	-1
θ_{10}	$\frac{1}{2}q(1 - q)^2$	0	$q - 1$	0	$q - 1$	1
θ_{13}	q^4	q	$-q$	q	$-q$	1

Table 5.1: Non-zero contributions for $\xi_G(2B, 3B, \mathcal{C})$

1. We have the following structure constants in G :

$$\xi_G(2B, 3A, \mathcal{C}) = 2 \quad (5.1.5)$$

$$\xi_G(2B, 3B, \mathcal{C}) = 2 \quad (5.1.6)$$

2. The corresponding structure constants in $G.2$ are:

$$\xi_{G.2}(2B, 3A, \mathcal{C}) = 1 \quad (5.1.7)$$

$$\xi_{G.2}(2B, 3B, \mathcal{C}) = 1 \quad (5.1.8)$$

Proof. 1. Examining the summation formula for structure constants (1.4.7) and the character table for $S_4(q)$ [50]¹ we see that most of the terms in the sum are zero. For those characters where there is a non-zero contribution, the values of the character on 3A and 3B agree, so it suffices to calculate the structure constant for 3B. The characters on which there is a non-zero contribution are given in Table 5.1, using notation from [50] for the characters.

¹The character table printed at the end of [50] contains a number of minor clerical errors which we discovered when testing whether the table was orthogonal. The calculations given in the main body of the text appear to be correct. The correct character table is available in CHEVIE [20].

Note that the character values depend on the residue class of q modulo 3 and 4. A straightforward calculation in each of the four cases gives the value of the structure constants for G in equations (5.1.5) and (5.1.6).

2. The group $G.2$ is isomorphic to $\text{SO}_5(q)$, and because the classes $2A, 2B, 3A, 3B$ and the various possibilities for \mathcal{C} correspond to different sets of eigenvalues, there can be no fusion of these classes in $G.2$, and each class remains the same size. Each character of G either gives rise to two characters in $G.2$, or fuses to give a single character of twice the degree. The effect of this is that the sum over the characters doubles. However, the group has doubled in size, so the overall effect on (1.4.7) is that the structure constant is halved.

This concludes the proof. ■

5.1.2 Standard pairs

Lemma 5.5 shows that to complete the definition of standard generators x, y for $S_4(q)$, we only need a condition on x and y which determines a particular conjugacy class \mathcal{C} of elements of order $(q^2 + 1)/2$. Because the classes $3A$ and $3B$ are not very easy to tell apart, we would like the condition to imply that y is a $3B$ element (given that it is an element of order 3). In other words, we want a condition $P_q(x, y)$ on elements x, y of $S_4(q)$ such that the following definition determines the pair up to automorphisms of $S_4(q)$:

Definition 5.6 *A standard pair for $S_4(q)$ is a pair (x, y) such that x is a $2B$ element, y has order 3, xy has order $(q^2 + 1)/2$ and the condition $P_q(x, y)$ holds. These conditions imply that y is a $3B$ element.*

To find a suitable condition $P_q(x, y)$, we found representatives for each class of triple of type $(2B, 3A/B, (q^2 + 1)/2)$. We know how many there are from the character table

and from Lemma 5.5, and we took a fingerprint for each triple to prove that they were mutually non-conjugate. We then selected a condition which singled out one of the classes of triples. Suitable $P_q(x, y)$ for small q are:

- $P_{11}(x, y) = "xyxy^2 \text{ has order } 10"$
- $P_{13}(x, y) = "xyxy^2 \text{ has order } 14"$
- $P_{17}(x, y) = "xyxy^2 \text{ has order } 5"$
- $P_{19}(x, y) = "xyxy^2 \text{ has order } 6"$

5.2 Black box algorithms for $S_4(q)$, $q = p^r$, $p > 3$ prime

If we are able somehow to distinguish the conjugacy classes 2A from 2B and 3A from 3B then we can use the following procedure to find standard generators for $S_4(q)$.

Algorithm 5.7 *To find standard generators of $S_4(q)$ with $q = p^r$, $p > 3$:*

1. Find random elements of the group, and power up those of even order to give elements x of order 2. Continue until x is a 2B element.
2. Find random elements of the group, and power up those whose order is divisible by 3 to give elements y of order 3. Continue until y is a 3B element.
3. Take random conjugates $z = y^g$ of y until xz has order $(q^2 + 1)/2$ and the condition $P_q(x, z)$ holds (this condition ensures that we have selected the correct automorphism class of semi-standard pairs: its auxiliary function of ensuring $y \in 3B$ is not required in this scenario).
4. Return x and z .

Steps 1 and 2 can be performed concurrently.

If we are not able to distinguish the classes $2A/B$ or $3A/B$, then a more complicated strategy is needed.

5.2.1 Distinguishing the two classes of involutions

We use the following lemma to help distinguish the two classes of involution in $S_4(q)$:

Lemma 5.8 *If z_1 and z_2 are elements of $S_4(q)$ in class $2A$, then $z_1 z_2$ has order at most q .*

Before we give the proof, we will need some extra information about the orthogonal group $\Omega_5(q)$, which is isomorphic to $S_4(q) = \text{PSp}_4(q)$ when q is odd.

Lemma 5.9 *Let $G = \Omega_5(q)$ acting on a 5-dimensional vector space V over $\text{GF}(q)$ equipped with a non-degenerate quadratic form $q : V \rightarrow \text{GF}(q)$. Then there are exactly 3 orbits of G on $V \setminus \{0\}$:*

- *isotropic vectors $v \in V$, where $q(v) = 0$;*
- *plus type vectors $v \in V$, where the quadratic form restricted to the hyperplane v^\perp has plus type; and*
- *minus type vectors $v \in V$, where the quadratic form restricted to the hyperplane v^\perp has minus type.*

The stabilizers in G of a vector of each of the three types have the following shapes:

$$\text{Stab}_G(\text{isotropic}) \cong q^3.\Omega_3(q) \cong q^3.L_2(q) \tag{5.2.1}$$

$$\text{Stab}_G(\text{plus type}) \cong q^3.\Omega_4^+(q) \cong 2.(L_2(q) \times L_2(q)) \tag{5.2.2}$$

$$\text{Stab}_G(\text{minus type}) \cong q^3.\Omega_4^-(q) \cong L_2(q^2) \tag{5.2.3}$$

Proof. See [29] ■

Proof (Lemma 5.8). As above, we will think of $S_4(q) = \text{PSp}_4(q)$ as $\Omega_5(q)$, so z_1 and z_2 are endomorphisms of a 5-dimensional space V over $\text{GF}(q)$ equipped with a non-degenerate quadratic form $q : V \rightarrow \text{GF}(q)$.

The (-1) -eigenspaces V_1 and V_2 of z_1 and z_2 (respectively) each have codimension 1. Thus the intersection $V_{12} = V_1 \cap V_2$ has codimension at most 2, and hence:

$$\dim(V_{12}) \geq 3 \quad (5.2.4)$$

For $i \in \{1, 2\}$, the element z_i inverts the subspace V_i , so $y = z_1 z_2$ acts trivially on V_{12} . The maximal isotropic subspaces of V have dimension $\frac{1}{2}(5 - 1) = 2$, so V_{12} contains a non-isotropic point v . Because y fixes v , it is contained in a subgroup $\Omega_4^\epsilon(q) \leq \Omega_5(q)$ for $\epsilon \in \{+, -\}$: we can think of y acting on a 4-dimensional orthogonal space W of type ϵ over $\text{GF}(q)$.

Suppose $\epsilon = -$. Then the maximum dimension of an isotropic subspace is $\frac{4}{2} - 1 = 1$, and so y must fix a non-isotropic point of W . Hence $y \in \Omega_3(q) \cong \text{L}_2(q)$, and thus cannot have order greater than q .

Otherwise, $\epsilon = +$. If y fixes a non-isotropic point in W , then once again $y \in \Omega_3(q)$ and hence has order at most q . So we may assume that y fixes a 2-dimensional isotropic subspace W' of W .

The number of isotropic vectors x in W is $(q + 1)(q^2 - 1)$, and the number of isotropic vectors y orthogonal to x is $2q(q - 1)$. The general orthogonal group $\text{GO}_4^+(q)$ is transitive on bases (x, y) of 2-dimensional isotropic subspaces, so by the orbit-stabilizer theorem, the order of the pointwise stabilizer of W' in $\text{GO}_4^+(q)$ is:

$$\frac{|\text{GO}_4^+(q)|}{(q + 1)(q^2 - 1)2q(q - 1)} = q \quad (5.2.5)$$

Now $\text{GO}_4^+(q) \geq \Omega_4^+(q)$, so y is in the pointwise stabilizer of W' in $\text{GO}_4^+(q)$. Hence it

has order dividing q . ■

Note that this lemma could also have been proved by considering the $\zeta(2A, 2A, -)$ structure constants.

We can use the lemma to give a one-sided test to tell whether an involution $z \in S_4(q)$ is in class $2B$ or not:

Algorithm 5.10 *To prove that an involution z is a $2B$ element:*

1. *Take a random conjugate z^g of z .*
2. *Calculate $w = zz^g$.*
3. *If w has order greater than q then z is in class $2B$ (by Lemma 5.8). Otherwise, go back to step 1.*

If the test has run several times without coming up with an element w of order greater than q , then z is probably in class $2A$.

5.2.2 Finding a $3B$ element

Let $\theta, \eta, \nu \in \text{GF}(q^4)$ be elements of order $q^2 - 1$, $q + 1$ and $q - 1$ respectively. The elements of projective order $(q^2 - 1)/2$ in $\text{Sp}_4(q)$ are of two types, distinguishable by their eigenvalues over $\text{GF}(q^4)$ [50]:

- elements with eigenvalues $\theta^i, \theta^{-i}, \theta^{qi}, \theta^{-qi}$ for some $i \in \mathbb{N}$;
- elements with eigenvalues $\eta^i, \eta^{-i}, \nu^j, \nu^{-j}$ for some $i, j \in \mathbb{N}$.

The first type (modulo scalars) powers up to classes $2B$ and $3A$, and the second type powers up to classes $2A$ and $3B$. The two types are equally likely to be found on a random search. Thus we can (probabilistically) find a $3B$ element in $S_4(q)$ as follows:

1. Find a random element t of order $(q^2 - 1)/2$.

2. Calculate $z = t^{(q^2-1)/4}$.
3. Use the test from section 5.2.1 to see whether z is a $2A$ or $2B$ element.
4. If z is a $2B$ element, go back to step 1.
5. If z is a $2A$ element, then $y = t^{(q^2-1)/6}$ is a $3B$ element.

Unfortunately, the test in section 5.2.1 cannot prove that an involution is in class $2A$, but if we perform enough iterations of the test, we have a high probability of being right. The small probability of being wrong should not be an issue, because our extra condition for standard generators is chosen so that it is never satisfied for a $3A$ element.

The finder is therefore as follows as follows:

Algorithm 5.11 (Finder for $S_4(q)$, $p > 3$) *To find standard generators for $S_4(q)$, $q = p^r$, $p > 3$:*

1. Find an element x in class $2B$, using the test in section 5.2.1.
2. Find (with high probability) an element y in class $3B$, using the procedure in section 5.2.2
3. Find a conjugate z of y such that xy has order $(q^2 + 1)/2$ and such that the extra condition (see section 5.1.1) is satisfied. If no such conjugate can be found, then the element y is probably a $3A$ element, so go back to step 2.
4. Return x and z .

For a checker, we only need to check that x is in class $2B$ (because the extra condition $P_q(x, y)$ then implies that y is in class $3B$). We can do this by using Algorithm 5.10.

5.3 Other groups $S_4(q)$

The only other groups in \mathcal{L}^G of the form $S_4(q)$ which did not have definitions for standard generators already are $S_4(8)$ and $S_4(9)$. These will be dealt with in sections 6.1.10 and 6.1.11 respectively.

Chapter 6

The remaining groups in \mathcal{L}^G

In this chapter, we will give standard generators for groups in \mathcal{L}^G where we did not think it worthwhile to give a ‘generic’ definition for a semi-standard pair. In each subsection below, we will provide a definition of standard generators for a group G using the techniques from Chapter 1.

If the standard finder (Algorithm 1.11) is not appropriate for the definition given, we will provide a finder. If the standard checker does not work (*i.e.* it is not sufficient to simply check the orders given in the definition), we will provide a checker. Table 6.1 gives the groups which need special finders and/or checkers. The approximate cost of finding each set of standard generators is given in Table 6.4 on page 129.

6.1 Groups with a readily available character table

Finding a definition for standard generators of a group G usually requires the calculation of certain structure constants (defined in section 1.4). If the character table of G is available, then the structure constants can be read off directly using (1.4.7). We will consider groups where we know the character table in this section. Subsequent sections will deal with the remaining cases, which require more substantial computation to complete the definitions of standard generators.

Group	Section	Page	Finder	Checker
$L_3(13)$	§6.1.1	p104	std	std
$L_4(4)$	§6.1.2	p106	6.4	(6.1.6)
$L_4(5)$	§6.1.3	p107	std	std
$L_5(3)$	§6.1.4	p108	std	(6.1.14)
$U_3(9)$	§6.1.5	p109	std	std
$U_3(13)$	§6.1.6	p110	std	std
$U_3(16)$	§6.1.7	p110	std	std
$U_4(4)$	§6.1.8	p111	6.11	(6.1.24)
$U_4(5)$	§6.1.9	p112	std	(6.1.28)
$U_5(4)$	§6.2.1	p120	std	(6.2.5)
$U_6(3)$	§6.2.2	p121	std	(6.2.7)
$U_8(2)$	§6.2.3	p122	std	(6.2.9)
$U_9(2)$	§6.3.1	p123	6.25	(6.3.9)
$S_4(8)$	§6.1.10	p113	6.13	(6.1.32)
$S_4(9)$	§6.1.11	p114	6.16	(6.1.34)
$S_6(5)$	§6.1.12	p116	std	(6.1.38)
$S_6(7)$	§6.3.2	p125	std	(6.3.13)
$S_8(3)$	§6.1.13	p117	std	(6.1.43)
$S_{10}(3)$	§6.3.3	p126	std	(6.3.15)
$O_{10}^+(2)$	§6.1.14	p118	std	(6.1.49)

Table 6.1: Finders and checkers

\mathcal{C}	$ C_G(\mathcal{C}) $	Taming set
1A	270178272	any
2A	8736	4, 6, 8, 12, 14, 26, 28, 52, 56
3A	144	6, 12
4AB	8736	8, 28, 52, 56
4C	48	12
6A	48	12
7ABC	56	14, 28
8AB	56	56
12ABCD	48	—
13A	8788	26, 52
13BCD	169	—
14ABC	56	28
26A	52	52
28ABCDEF	56	56
52AB	52	—
56AB...L	56	—
61AB...T	61	—

Table 6.2: Conjugacy classes in $L_3(13)$

6.1.1 Linear group $L_3(13)$

Let $G = L_3(13)$. To illustrate the method, we will show the working in some detail for this group.

Although **GAP** does not currently have the character table of the group G , the generic character table for $L_3(q)$ with $q \equiv 1 \pmod{3}$ is available in **CHEVIE** [20]. Using the centralizer orders in this table and some calculations in **GAP** to establish element orders, we obtained the information in Table 6.2 about the conjugacy classes of G .

For a semi-standard pair, we need to find classes which are both easy to find, have large centralizers, and have a chance of generating G . The classes 2A, 4A, 4B and 13A have large centralizers, but we cannot generate the group with a pair from these classes.

We decide on 2A for the class of the first generator. We can certainly $(2A, 13BCD)$ -

No.	$o((xy)^2y)$	$o((xy)^3y)$	$o((xy)^4y)$	$o((xy)^3yxy^2)$
1	4	28	61	7
2	6	61	61	7
3	12	14	28	7
4	12	28	28	14
5	12	61	56	14
6	13	28	61	14
7	14	7	8	26
8	14	28	61	3
9	14	56	61	7
10	26	56	12	3

Table 6.3: Fingerprints for $(2, 3, 61)$ -pairs of $L_3(13)$

generate the group, but it may be hard to distinguish between classes $13BCD$ and $13A$. The centralizer is only slightly smaller with $3A$, and $3A$ is tame. Experiments with GAP show that possible $(2A, 3A)$ -targets¹ are 56 and 61. There is little to choose between these targets, but the structure constant calculations are slightly simpler with 61. We define:

Definition 6.1 *A semi-standard pair for $L_3(13)$ is a pair (x, y) with x of order 2, y of order 3 and xy of order 61.*

Another calculation with CHEVIE yields:

$$\xi(2A, 3A, \mathcal{C}) = 3 \quad (6.1.1)$$

for any of the 20 classes \mathcal{C} containing elements of order 61. The outer automorphism group of G is S_3 , and the inverse-transpose automorphism fuses pairs of classes. Thus there are only 10 classes \mathcal{C}' containing elements of order 61 in $\text{Aut}(G)$ and for these classes we have:

$$\xi_{\text{Aut}(G)}(2A, 3A, \mathcal{C}') = 1 \quad (6.1.2)$$

¹Recall that a *target* k is an integer such that we can find a suitable pair (x, y) satisfying $o(xy) = k$ which generates G .

Providing each semi-standard pair generates the whole of G , there are exactly 10 fingerprints to find. We performed a random search in GAP and found the 10 fingerprints given in Table 6.3. We verified that each of these 10 semi-standard pairs generates G , which proves that there are no more classes of semi-standard pairs to find. Examining the table shows that we can pick out a single class of pairs (row 1 in the table) by adding the supplementary condition $o(xyxy^2) = 4$. We can thus supply the following definition of standard generators:

Definition-Theorem 6.2 *Standard generators for $L_3(13)$ are a pair (x, y) such that x has order 2, y has order 3, xy has order 61 and $xyxy^2$ has order 4.*

These generators are completely tame (see Table 6.2), so we can use Algorithm 1.11 to find them. The associated probabilities are:

$$p_1 = 0.594, \quad p_2 = 0.111, \quad p_{12} = 0.104. \quad (6.1.3)$$

We have:

$$EX = 9.02, \quad EY = 35.80, \quad (6.1.4)$$

and so

$$ET = 44.82 \quad (6.1.5)$$

6.1.2 Linear group $L_4(4)$

The group $G = L_4(4)$ can be $(2B, 3D)$ -generated, but the class $4A$ is tame ($T(4A) = \{12\}$) with larger centralizer than $3D$, so we prefer $(2B, 4A)$ -generators. The possible targets k for $(2B, 4A)$ are 30, 63 and 85, but for $k = 30$ we do not need any supplementary conditions.

Definition-Theorem 6.3 *Standard generators for $L_4(4)$ are a pair (x, y) such that x is in $2B$, y is in $4A$ and xy has order 30.*

The class $2B$ is not tame, but there are only two classes of involutions in G . If we power up an element of order 2, 4, 6 or 10 found in a random search, the probability of getting a $2B$ element is about 0.47. Now $\xi(2A, 2A, k) = 0$ for $k > 5$, and about 74% of the time, the product of a $2B$ element with another $2B$ element has order larger than 5. Thus we can use the following finder:

Algorithm 6.4 *Finder for $L_4(4)$:*

1. Find an element of order 2, 4, 6 or 10 and power it up to give an involution x .
2. Look for a random element z such that $[x, z]$ has order greater than 5. If none can be found after two attempts, go back to step 1. Otherwise, we know x is a $2B$ element.
3. Find an element of order 12 and power it up to give an element t in class $4A$.
4. Find a conjugate y of t such that xy has order 30.
5. Return x and y .

For a checker, it is not sufficient to check the orders. Structure constants and fingerprinting show that there are 14 classes of $(2B, 4B, 30)$ -pairs up to automorphism, and a single class of $(2A, 4B, 30)$ -pairs up to automorphism. We get:

$$L_4(4) \approx \langle \langle x, y \mid \mathbf{std}, o(xyxy^2) = 21 \rangle \rangle \quad (6.1.6)$$

6.1.3 Linear group $L_4(5)$

The group $G = L_4(5)$ can be $(2A, 3B)$ -generated. The class $2A$ is completely tame, and $3B$ is tame with

$$T(3B) = \{24, 30\} \quad (6.1.7)$$

The possible targets are 31 and 39. There are 6 conjugacy classes \mathcal{C} of order 39 in G , and for each class we have:

$$\zeta(2A, 3B, \mathcal{C}) = \frac{14}{3} = 4 \left(1 + \frac{1}{6}\right) \quad (6.1.8)$$

The outer automorphism group is D_8 , and there are just 3 classes \mathcal{C}' of elements of order 39 in $\text{Aut}(G)$. Thus in $\text{Aut}(G)$ we have:

$$\zeta_{\text{Aut}(G)}(2A, 3B, \mathcal{C}') = 1 + \frac{1}{6} \quad (6.1.9)$$

We found 6 different fingerprints; 3 of which generated G , and 3 of which generated a group which has centralizer of order 6 in $\text{Aut}(G)$ (order 3 in G), so this is a complete set of fingerprints.

Definition-Theorem 6.5 *Standard generators for $L_4(5)$ are a pair (x, y) such that x has order 2, y is in class $3B$, xy has order 39 and $xyxy^2$ has order 20.*

There are only two classes containing elements of order 3, and since there are no $(2A, 3A, 39)$ -pairs, the standard checker works for $L_4(5)$.

6.1.4 Linear group $L_5(3)$

The group $G = L_5(3)$ can be generated by the tame classes $2A$ and $5A$:

$$T(2A) = \{10, 16, 18, 20, 40, 80\} \quad (6.1.10)$$

$$T(5A) = \{5, 10, 20, 40, 80\} \quad (6.1.11)$$

Possible targets k are given by:

$$\{11, 20, 24, 39, 40, 52, 104, 121\} \quad (6.1.12)$$

The cases $k = 11$ and $k = 20$ are equally easy, but we chose $k = 11$. There are two conjugacy classes containing elements of order 11, and they fuse in the automorphism group $L_5(3).2$, with:

$$\xi_{L_5(3).2}(2A, 5A, 11AB) = 1 \quad (6.1.13)$$

Definition-Theorem 6.6 Standard generators for $L_5(3)$ are a pair (x, y) such that x is in class $2A$, y has order 5 and xy has order 11.

We give the following two semi-presentations:

$$L_5(3) \approx \langle \langle x, y \mid \mathbf{std}, o(xy^2) = 121, o(xy^2xy^3) = 3 \rangle \rangle \quad (6.1.14)$$

$$L_5(3) \approx \langle \langle x, y, (z) \mid \mathbf{std}, o(z) = 80, o(xz^{40}) = 3; z = (xy)^3xy^3 \rangle \rangle \quad (6.1.15)$$

6.1.5 Unitary group $U_3(9)$

The group $G = U_3(9)$ can be $(2, 3)$ -generated, but the elements of order 3 can be difficult to tell apart. It is much simpler to use $(2, 6)$ -generators. The classes $2A$ and $6A$ are completely tame:

$$T(2A) = \{2, 4, 6, 8, 10, 16, 20, 30, 40, 80\} \quad (6.1.16)$$

$$T(6A) = \{6, 30\} \quad (6.1.17)$$

The possible $(2A, 6A)$ -targets k are given by:

$$k \in \{10, 15, 20, 30, 40, 73, 80\} \quad (6.1.18)$$

Not all the $(2, 6, 10)$ -pairs are generating pairs, so we choose to $(2, 6, 15)$ -generate G . There are 4 conjugacy classes of order 15, and for each class \mathcal{C} we have:

$$\xi(2A, 6A, \mathcal{C}) = 1 \quad (6.1.19)$$

The classes of order 15 fuse in $\text{Aut}(G)$, and so all the $(2, 6, 15)$ -pairs are equivalent.

Definition-Theorem 6.7 *Standard generators for $U_3(9)$ are a pair (x, y) such that x has order 2, y has order 6 and xy has order 15.*

6.1.6 Unitary group $U_3(13)$

The group $G = U_3(13)$ has a single conjugacy class of involutions and a single class of elements of order 3. It can be $(2, 3, k)$ -generated with:

$$k \in \{14, 21, 42, 56, 84, 91, 157, 168, 182\} \quad (6.1.20)$$

Examining the character table shows that not all the $(2, 3, 14)$ -pairs generate the group, but all the other possibilities would make suitable choices for semi-standard generators. We choose $(2, 3, 42)$, which gives 3 classes of semi-standard pairs.

Definition-Theorem 6.8 *Standard generators for $U_3(13)$ are a pair (x, y) of elements of $U_3(13)$ such that x has order 2, y has order 3, xy has order 42 and $xyxyxy^2$ has order 7.*

6.1.7 Unitary group $U_3(16)$

The group $G = U_3(16)$ has a single conjugacy class of involutions and a single conjugacy class of elements of order 3, and G can be $(2, 3, k)$ -generated with:

$$k \in \{17, 51, 85, 241, 255\} \quad (6.1.21)$$

We will define a semi-standard pair to be $(2, 3, 17)$; although in fact not all such pairs generate G . There are 56 classes \mathcal{C} containing elements of order 17 in G , and:

$$\zeta(2A, 3A, \mathcal{C}) = \begin{cases} 0 & \text{for 16 classes } \mathcal{C} \\ 1 & \text{for 32 classes } \mathcal{C} \\ 1/17 & \text{for 8 classes } \mathcal{C} \end{cases} \quad (6.1.22)$$

The 32 classes with $\zeta = 1$ fuse into 4 classes in the automorphism group, and the 8 classes with $\zeta = 1/17$ fuse into 2 classes. Thus there are 6 equivalence classes of semi-standard pairs, 4 of which generate the group. We found fingerprints for each class.

Definition-Theorem 6.9 Standard generators for $U_3(16)$ are a pair (x, y) such that x has order 2, y has order 3, xy has order 17 and $xyxy^2$ has order 5.

6.1.8 Unitary group $U_4(4)$

The group $G = U_4(4)$ can be $(2, 3)$ -generated, but we prefer $(2, 4)$ -generation because the pairs are slightly easier to find. We have classes $2A$ (tame with $T(2A) = \{20, 30\}$), $2B$ (non-tame), $4A$ (tame with $T(4A) = \{20\}$) and $4B$ (non-tame). The only combinations which generate G are $(2B, 4A)$ and $(2A, 4B)$. The second type is slightly easier to find because of the larger taming set for $2A$.

The group is $(2A, 4B, k)$ generated for $k \in \{20, 30, 51, 65\}$. There are 4 classes \mathcal{C} of order 20 in G , and:

$$\zeta(2A, 4B, \mathcal{C}) = 1 \quad (6.1.23)$$

The classes \mathcal{C} fuse to a single class \mathcal{C}' under the action of the group of field automorphisms of $GF(16)$ (cyclic of order 4).

Definition-Theorem 6.10 Standard generators for $U_4(4)$ are a pair (x, y) such that x is in $2A$, y is in $4B$ and xy has order 20.

The class $4B$ is a lot larger than the class $4A$, but we still need some way of telling the classes apart. We check that the structure constant $\tilde{\zeta}(2A, 4A, k)$ is zero for $k > 15$, so the following black box algorithm works:

Algorithm 6.11 *Finder for $U_4(4)$:*

1. Find an element of order 20 or 30 and power it up to an involution x .
2. Find an element t of order 4 (not by powering up).
3. Look for a conjugate y of t such that xy has order 20. If all the orders of xy seem to be 15 or less, t is probably in the wrong class, so go back to step 2.
4. Return x and y .

The standard checker is not sufficient for G , as there are $(2B, 4A, 20)$ and $(2B, 4B, 20)$ pairs. However, there are no $(2A, 4A, 20)$ pairs, so it suffices to check that x is in $2A$. We can do this by powering up xy of order 20 to give another $2A$ element z , and checking the order of xz . We have:

$$U_4(4) \approx \langle \langle x, y \mid \mathbf{std}, o(x(xy)^{10}) = 3 \rangle \rangle \quad (6.1.24)$$

6.1.9 Unitary group $U_4(5)$

The group $G = U_4(5)$ can be $(2A, 4B, k)$ -generated for:

$$k \in \{9, 24, 63\} \quad (6.1.25)$$

We have:

$$\tilde{\zeta}_G(2A, 4B, \mathcal{C}) = 2 \quad (6.1.26)$$

for the two classes \mathcal{C} of elements of order 9. The group $\text{Out}(G)$ has order 4, and the two classes fuse to give a single class \mathcal{C}' under the field automorphism $x \mapsto x^5$ of $\text{GF}(5^2)$.

Hence we have:

$$\zeta_{\text{Aut}(G)}(2A, 4B, \mathcal{C}') = 1 \quad (6.1.27)$$

and all $(2A, 4B, 9)$ -pairs are equivalent.

Definition-Theorem 6.12 *Standard generators of $U_4(5)$ are a pair (x, y) such that x is in $2A$, y is in $4B$ and xy has order 9.*

These generators are tame, with $T(2A) = \{8, 24, 30, 60\}$ and $T(4B) = \{8, 24, 60\}$, so we can use Algorithm 1.11 to find generators. In fact, class $4B$ squares to $2A$, so we can find representatives of both classes together.

The standard checker is not sufficient; we need to check the classes of x and y . This can easily be done by verifying that x and y^2 are $2A$ elements, as the other classes of 4-elements square to $2B$. We find a $2A$ element z by powering up an element with order in $T(2A)$, and search for elements g and h such that x^8z and $(y^2)^hz$ have odd order. We have:

$$\begin{aligned} U_4(5) \approx \langle \langle x, y, (z, w) \mid \mathbf{std}, o(w) = 30, o(x^y z) = 5, o((y^2)^x z) = 5; \\ w = xyxy^2; z = w^{15} \rangle \rangle \end{aligned} \quad (6.1.28)$$

6.1.10 Symplectic group $S_4(8)$

Let $G = S_4(8)$. The outer automorphism group of G is cyclic of order 6, generated by elements ϕ (the Frobenius automorphism of order 3) and γ (the extraordinary graph automorphism of order 2 for the Dynkin diagram C_2).

The conjugacy classes $2A$ and $2B$ of $S_4(8)$ fuse under the element γ , and we have:

$$T(2A/B) = \{6, 14, 18\} \quad (6.1.29)$$

Moreover there is a unique class of elements of order 5:

$$T(5A) = \{5, 65\} \quad (6.1.30)$$

The group can be $(2A, 5A, 7)$ -generated, and if \mathcal{C} is a class containing elements of order 7, then we have:

$$\xi_G(2A, 5A, \mathcal{C}) = \begin{cases} 1 & \text{if } \mathcal{C} \in \{7G, 7H, 7I\} \\ 0 & \text{otherwise} \end{cases} \quad (6.1.31)$$

The classes $7G$, $7H$ and $7I$ fuse under the Frobenius automorphism ϕ . Therefore our definition of standard generators is:

Definition-Theorem 6.13 *Standard generators for $S_4(8)$ are a pair (x, y) such that x is in class $2A$ or $2B$, y has order 5 and xy has order 7.*

The standard checker is not sufficient; we need to make sure that $x \notin 2C$. We use Lemma 1.14 and the fact that all elements of order 18 power up to $2A/B$. We have:

$$S_4(8) \approx \langle \langle x, y, (z) \mid \mathbf{std}, o(z) = 18, o(xz^9) = 9; z = xyxyxyxy^4 \rangle \rangle \quad (6.1.32)$$

6.1.11 Symplectic group $S_4(9)$

The group $G = S_4(9)$ has three classes of elements of order 4:

- class $4A$ squaring to $2B$ (centralizer order 2880);
- class $4B$ squaring to $2A$ (centralizer order 2880);

- class 4C squaring to 2B but not powerable² itself (centralizer order 64).

Definition 6.14 A semi-standard pair for $S_4(9)$ is a pair (x, y) of elements of $S_4(9)$ where x is in class 2B, y is in class 4B and xy has order 41.

Lemma 6.15 Each semi-standard pair for $S_4(9)$ generates $S_4(9)$.

Proof. Let $G = S_4(9)$ and suppose $H = \langle x, y \rangle$ is a proper subgroup of G . As with Theorem 5.4, we work in the double cover $\tilde{G} = \text{Sp}_4(9)$ and write \tilde{t} to denote a preimage of $t \in G$ in \tilde{G} . As before, we have a Singer cycle $\tilde{x}\tilde{y}$ (or $-\tilde{x}\tilde{y}$), so by Lemma 5.3, the group $\tilde{H} = \langle -1, \tilde{x}, \tilde{y} \rangle$ is contained in a subgroup $\tilde{L} \cong \text{Sp}_2(81).2 \cong \text{SL}_2(81).2$. In the natural action of this subgroup on a 2-dimensional vector space over $\text{GF}(81)$, the elements of order 4 do not fix any vectors. However, all 4B elements of \tilde{G} fix a 2-dimensional submodule of the natural 4-dimensional $\text{GF}(9)\tilde{G}$ -module. This is a contradiction. ■

For any class \mathcal{C} of elements of order 41, we have the structure constant:

$$\zeta_{S_4(9)}(2B, 4B, \mathcal{C}) = 2 \tag{6.1.33}$$

This becomes 1 in the automorphism group: $\text{Out}(G)$ has order 4, but there is some fusing of the conjugacy classes \mathcal{C} . In the automorphism group, there are 5 classes of triples to consider. We found fingerprints for all 5 (and also for the $(2B, 4A, \mathcal{C})$ -triples), which enabled us to define:

Definition-Theorem 6.16 Standard generators for $S_4(9)$ are a pair (x, y) of elements of $S_4(9)$ where x is in class 2B, y is in class 4B, xy has order 41 and $xyxy^3$ has order 5.

²A group element x is *powerable* if an element of greater order powers up to x . The definition extends naturally to conjugacy classes.

If y is in class $4A$, then the conditions $o(xy) = 41$ and $o(xyxy^3) = 5$ cannot hold simultaneously. Thus we get the following black box algorithm:

Algorithm 6.17 *Finder for $S_4(9)$:*

1. Find a random element of even order and power it up to give an involution x .
2. Test whether x is a $2B$ element using Lemma 5.8. If it cannot be proved to be in $2B$, go back to step 1.
3. Find an element u of order o satisfying $o > 4, 4 \mid o$.
4. Test whether $u^{o/2}$ is a $2A$ element using Lemma 5.2.1. If it is (probably) a $2A$, then let $y = u^{o/4}$. This is probably a $4B$ element, although it may be a $4A$ element. If $u^{o/2}$ is a $2B$ element, go back to step 3.
5. Find a conjugate z of y such that xz has order 41 and $xzxz^3$ has order 5. If no such conjugate can be found, we may have found a $4A$ element, so go back to step 3.
6. Return x and z .

For a checker, we only need to show $x \in 2B$ (the class of y is determined by the standard relations). We can do this with Lemma 5.8:

$$S_4(9) \approx \langle \langle x, y \mid \mathbf{std}, o(x^{yxy}x) = 40 \rangle \rangle \quad (6.1.34)$$

6.1.12 Symplectic group $S_6(5)$

The group $G = S_6(5)$ is $(2A, 8A)$ -generated, and the two classes are tame:

$$T(2A) = \{4, 8, 12, 20, 24, 26, 30, 40, 52, 60, 78, 120, 130\} \quad (6.1.35)$$

$$T(8A) = \{40, 120\} \quad (6.1.36)$$

There are a number of possible targets k , but a convenient choice is $k = 9$. There is a single conjugacy class of order 9, and we have:

$$\zeta(2A, 8A, 9A) = 4 \quad (6.1.37)$$

These 4 classes of pairs fuse to just 2 in $\text{Aut}(G)$, and so there are 2 fingerprints to find.

Definition-Theorem 6.18 *Standard generators of $S_6(5)$ are a pair (x, y) such that x is in $2A$, y is in $8A$, xy has order 9 and xy^2 has order 31.*

For a checker, we can check $x \in 2A$ using Lemma 1.8.1 as usual. We can then find the 31 fingerprints for the $(2A, 8B, 9)$ -pairs. It turns out that the conditions we already have are enough to show $y \in 8A$.

$$S_6(5) \approx \langle \langle x, y, (z) \mid \mathbf{std}, o(z) = 52, o(x^{y^5} z^{26}) = 3; z = xy^3 \rangle \rangle \quad (6.1.38)$$

6.1.13 Symplectic group $S_8(3)$

The group $G = S_8(3)$ can be $(2B, 4D)$ -generated. Possible targets k are given by:

$$k \in \{14, 15, 18, 20, 21, 30, 36, 39, 40, 41, 42, 45, 60, 78, 90\} \quad (6.1.39)$$

For $k = 14$, there is a unique target class $14B$, and we have:

$$\zeta(2B, 4D, 14B) = 2 \quad (6.1.40)$$

But since $\text{Aut}(G) = G.2$, there is a unique automorphism class of $(2B, 4D, 14)$ -pairs.

Definition-Theorem 6.19 *Standard generators for $S_8(3)$ are a pair (x, y) such that x is in $2B$, y is in $4D$ and xy has order 14.*

These generators are tame, with:

$$T(2B) = \{24, 72, 90\} \quad (6.1.41)$$

$$T(4D) = \{84\} \quad (6.1.42)$$

so we can use Algorithm 1.11 to find them.

For a checker, observe that the classes of elements of order 4 which power to $2A$ are $4A$, $4C$ and $4D$, and $\zeta(2B, 4A, 14) = \zeta(2B, 4C, 14) = 0$. Hence we only need to check $x \in 2B$ and $y^2 \in 2A$. We can do both using Lemma 1.8.1. We have:

$$\begin{aligned} S_8(3) \approx \langle \langle x, y, (z, w) \mid \mathbf{std}, o(z) = 24, o(x^y z^{12}) = 9, o(w) = 52, \\ o((y^2)^{xyxy} w^{26}) = 3; z = xy^3 xyxy^2 xy, w = xyxy^2 \rangle \rangle \end{aligned} \quad (6.1.43)$$

6.1.14 Orthogonal group $O_{10}^+(2)$

The group $G = O_{10}^+(2)$ has two classes of order 20, only one of which ($20A$) is tame. The group can be $(2A, 20A, k)$ -generated with:

$$k \in \{21, 30, 31, 45, 51, 60\} \quad (6.1.44)$$

There are 3 classes of elements of order 21, but classes $21A$ and $21B$ fuse in $\text{Aut}(G) = O_{10}^+(2):2$. We have:

$$\zeta(2A, 20A, 21A/B) = 1 \quad (6.1.45)$$

$$\zeta(2A, 20A, 21C) = 0 \quad (6.1.46)$$

Thus there is only one $(2A, 20A, 21)$ pair up to automorphisms of G .

Definition-Theorem 6.20 Standard generators of $O_{10}^+(2)$ are a pair (x, y) such that x is

in $2A$, y is in $20A$ and xy has order 21.

These generators are tame, with:

$$T(2A) = \{24, 42, 60\} \quad (6.1.47)$$

$$T(20A) = \{60\} \quad (6.1.48)$$

so we can use Algorithm 1.11 to find them.

To check them, note that the only conjugacy class of elements of order 20 which powers to $2A$ is $20A$. Thus we need to show x and y^{10} are both in $2A$, which we can do using Lemma 1.8.1. We have:

$$\begin{aligned} O_{10}^+(2) \approx \langle \langle x, y, (z) \mid \mathbf{std}, o(z) = 42, o(x^y z^{21}) = 3, \\ o((y^{10})^{xy} z^{21}) = 3; z = xyxy^8 \rangle \rangle \end{aligned} \quad (6.1.49)$$

6.2 Groups whose character table is not available

We do not have character tables for the remaining groups, so we must proceed differently.

Let G be a group. If we can find conjugacy classes $\mathcal{C}_1, \mathcal{C}_2$ of G such that \mathcal{C}_1 is small and G is $(\mathcal{C}_1, \mathcal{C}_2)$ -generated, then we can find a definition for standard generators as follows. Let y be an arbitrary element of \mathcal{C}_2 . Then $C_G(y)$ is a subgroup of G and it acts on \mathcal{C}_1 by conjugation. Each orbit of this action corresponds to a conjugacy class of $(\mathcal{C}_1, \mathcal{C}_2)$ -pairs in G . Enumerating these orbits requires a lot of computation, which is why we require \mathcal{C}_1 to be small.

Some of the groups in this section are permutation groups of large degree, and calculating the orbits of this action directly requires too much memory. Instead, we pick elements g of \mathcal{C}_1 at random, and calculate an invariant (a *hash key* in the parlance

of computer science):

$$I(g) = \sum_{n \in \Psi} n \times n^g \quad (6.2.1)$$

where Ψ is a fixed but arbitrary subset of Ω (and G is a permutation group acting on Ω). Note that by convention the elements of Ω are integers, so the multiplication in (6.2.1) takes place in \mathbb{N} .

If we ever see a value for $I(g)$ which we have already seen, we discard g and try again. Otherwise, we record $I(h)$ for all h in the $C_G(\mathcal{C}_2)$ -orbit Δ of g , and perform any other calculations with the orbit that we require. We continue doing this until:

$$\sum |\Delta| = |\mathcal{C}_1| \quad (6.2.2)$$

where the sum is over orbits Δ that we have seen so far.

The formula (6.2.1) for the invariant is of course arbitrary—the point is that it is reasonably fast to calculate and we are unlikely to find two elements g with the same value of $I(g)$. It is even more unlikely that there are two orbits Δ_1 and Δ_2 such that

$$\{I(g) : g \in \Delta_1\} \subseteq \{I(g) : g \in \Delta_2\} \quad (6.2.3)$$

which is the only circumstance in which our strategy could fail. If we ever suspected that this had occurred, we could always choose a different set Ψ and start again.

6.2.1 Unitary group $U_5(4)$

The group $G = U_5(4)$ has two classes of elements of order 2. The smallest non-trivial conjugacy class is $2A$, with size 52275. The next smallest is $2B$, with size 13591500, which is too large to handle easily. We therefore choose $2A$ as the class for our first

generator. The class $2A$ is tame, with:

$$T(2A) = \{8, 12, 20, 26, 30\} \quad (6.2.4)$$

There is a unique class $12A$ of elements of order 12, and the group is $(2A, 12A, 41)$ -generated. Moreover, $1/48$ of the group is in this class, so it is easy to find elements in it (even though it is not a powerable class). The centralizer $C_G(12A) \simeq C_{12} \times C_4$ acts on $2A$ by conjugation. A computer search taking just under 4 minutes showed that there are 1355 orbits, 200 of which correspond to $(2A, 12A, 41)$ -pairs. The outer automorphism group of G is $5:4$, and so there are just 10 fingerprints to find.

Definition-Theorem 6.21 *Standard generators for $U_5(4)$ are a pair (x, y) such that x is in $2A$, y has order 12, xy has order 41 and xy^2 has order 15.*

To check the generators, we only need to show $x \in 2A$, and since we know $y^6 \in 2A$, we can use Lemma 1.8.1.

$$U_5(4) \approx \langle \langle x, y \mid \mathbf{std}, o(xy^6) = 5 \rangle \rangle \quad (6.2.5)$$

6.2.2 Unitary group $U_6(3)$

The two smallest non-trivial conjugacy classes of the group $G = U_6(3)$ are the classes $2A$ and $3A$ of sizes 44226 and 44408 respectively. The next smallest is 36420111, which is too many points to handle comfortably. We therefore chose $\mathcal{C}_1 = 2A$, which is a tame class with:

$$T(2A) = \{122\} \quad (6.2.6)$$

The transvection class $3A$ would have worked almost as well.

We then considered classes \mathcal{C}_2 such that G can be $(2A, \mathcal{C}_2)$ -generated and such that

\mathcal{C}_2 can be easily found. The class $15A$ is the most straightforward possibility: it is a completely tame class, and it can be powered from elements of orders 30, 60 and 120.

To finish the definition, we looked for orbits of $C_G(15A)$ on the class $2A$. It took about $2\frac{1}{2}$ minutes and about 24000 random trials to find the 492 orbits. Of these, 48 correspond to $(2A, 15A, 91)$ -pairs. The group $\text{Out}(G)$ has order 4, and so there were only 12 equivalence classes of pairs.

Definition-Theorem 6.22 Standard generators for $U_6(3)$ are a pair (x, y) such that x is in $2A$, y has order 15, xy has order 91 and xy^2 has order 14.

For the checker, we can use Lemma 1.8.1 again:

$$U_6(3) \approx \langle \langle x, y, (z) \mid \mathbf{std}, o(z) = 122, o(x^{y^5}xyz^{61}) = 3; z = xy^3xy \rangle \rangle \quad (6.2.7)$$

6.2.3 Unitary group $U_8(2)$

The smallest non-trivial conjugacy class of the group $G = U_8(2)$ is the class $2A$. It has size 10965, and it is the only tame class of involutions:

$$T(2A) = \{14, 22, 42, 44, 66, 72, 126, 132\} \quad (6.2.8)$$

The group G can be $(2A, 14A, 85)$ -generated, and the class $14A$ is completely tame. The centralizer $C_G(14A) \simeq C_{14} \times C_9$ acts on $2A$ by conjugation with 133 orbits, 8 of which correspond to $(2A, 14A, 85)$ -pairs. The automorphism group is $U_8(2):2$, and there are just 4 equivalence classes of such pairs.

Definition-Theorem 6.23 Standard generators for $U_8(2)$ are a pair (x, y) such that x is in $2A$, y has order 14, xy has order 85 and xy^2 has order 18.

We have:

$$U_8(2) \approx \langle \langle x, y \mid \mathbf{std}, o(xy^7) = 3 \rangle \rangle \quad (6.2.9)$$

6.3 Groups where calculating conjugacy classes is impractical

The remaining groups are too large to compute a complete set of conjugacy classes. We must usually be content with finding a subset of each taming set.

6.3.1 Unitary group $U_9(2)$

Let $G = U_9(2)$, and let $GF(4) = \{0, 1, \omega, \omega^2\}$. There are 4 conjugacy classes of involutions in G . These classes correspond to matrices in $SU_9(2)$ with the canonical forms M_i , $1 \leq i \leq 4$: where:

$$M_i = \underbrace{T_2 \oplus \cdots \oplus T_2}_{i \text{ times}} \oplus \underbrace{I_2 \oplus \cdots \oplus I_2}_{4-i \text{ times}} \oplus I_1 \quad (6.3.1)$$

and

$$T_2 = \begin{pmatrix} 0 & \omega \\ \omega^2 & 0 \end{pmatrix}, \quad I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad I_1 = \begin{pmatrix} 1 \end{pmatrix} \quad (6.3.2)$$

The orders of their centralizers are:

$$|C_G(2A)| = 2^{36} \cdot 3^8 \cdot 5 \cdot 7 \cdot 11 \cdot 43 \quad (6.3.3)$$

$$|C_G(2B)| = 2^{35} \cdot 3^6 \cdot 5 \cdot 11 \quad (6.3.4)$$

$$|C_G(2C)| = 2^{33} \cdot 3^6 \quad (6.3.5)$$

$$|C_G(2D)| = 2^{30} \cdot 3^4 \cdot 5 \quad (6.3.6)$$

Thus the class $2A$ is the smallest class of involutions, containing just 43605 elements. Because any element powering up to an involution is in the centralizer of that involution, any element of even order which is divisible by 7 or 43 must power up to $2A$.

Thus:

$$T(2A) \supseteq \{14, 28, 42, 84, 86, 126\} \quad (6.3.7)$$

The group G can be $(2A, 17)$ -generated, and for any element t of order 17, we have

$$C_G(t) \simeq C_{17} \times C_5 \quad (6.3.8)$$

However, there are two conjugacy classes $17A$ and $17B$ of order 17 (this can be verified by observing that the Sylow 17-subgroup S is cyclic, and if $S = \langle s \rangle$, then s is conjugate in G to s^2 but not to s^3). Examining fingerprints shows that these classes do not fuse in the automorphism group.

After about $3\frac{1}{2}$ minutes of searching, we found the 513 orbits of $C_G(17A)$ on the class $2A$, of which 36 correspond to $(2A, 17A, 57)$ pairs. The group $\text{Out}(G)$ is isomorphic to S_3 , so there are only 6 equivalence classes of such pairs. Similarly, there are 6 equivalence classes of pairs with the class $17B$ instead of $17A$. We will use fingerprints to tell the two classes apart.

Definition-Theorem 6.24 *Standard generators for $U_9(2)$ are a pair (x, y) such that x is in $2A$, y has order 17, xy has order 57 and xy^2 has order 17 (or equivalently, $xyxy^2$ has order 84). Either condition fixes the class of y to be $17A$.*

The following algorithm can find standard generators.

Algorithm 6.25 *Finder for $U_9(2)$:*

1. Find an element of order 14, 28, 42, 84, 86 or 126 and power it up to an involution x in class $2A$.
2. Find an element of order 17 or 85 and power it up (if necessary) to give an element y of order 17.

3. Look for a conjugate y of t such that xy has order 57.
4. If $xyxy^2$ has order 17, 36, 72, 99 or 132 then replace t by t^3 and go back to step 3 (we should only have to do this once, if at all).
5. If $xyxy^2$ has order 45, 60 or 126 then go back to step 3. (Otherwise, $xyxy^2$ must have order 84.)
6. Return x and y .

To check the generators, we use Lemma 1.8.1 as usual: We have:

$$\mathrm{U}_9(2) \approx \langle \langle x, y, (z) \mid \mathbf{std}, o(z) = 28, o(xz^{14}) = 3; z = xy^7 \rangle \rangle \quad (6.3.9)$$

6.3.2 Symplectic group $S_6(7)$

Let $G = S_6(7)$. The only conjugacy classes of G which are small enough to be amenable to our orbit-counting approach are those of the transvections. There are two conjugacy classes of transvections $7A$ and $7B$, and these are swapped by the map $g \mapsto g^3$. These classes fuse to a single class $7A/B$ in the automorphism group.

The Sylow 5-subgroup of $S_6(7)$ is C_{25} , and all elements of order 5 in G are conjugate. By examining $C_G(5A)$, a group of order 8400, we can prove that there are two conjugacy classes of order 35 in G , each of which powers up to a transvection. Hence we have:

$$\mathrm{T}(7A/B) \supseteq \{35, 70, 175, 350\} \quad (6.3.10)$$

There is a single conjugacy class $9A$ of elements of order 9, and we have:

$$\mathrm{T}(9A) \supseteq \{9, 171\} \quad (6.3.11)$$

The group can be $(7A, 9, 57)$ -generated. The centralizer of a $9A$ element is $C_G(9A) \simeq$

$C_9 \times C_{19}$. After about $2\frac{1}{2}$ minutes of computer searching, we found the 344 orbits of the action of $C_G(9A)$ on the conjugacy class $7A$, 12 of which correspond to $(7A, 19, 57)$ -pairs.

Definition-Theorem 6.26 *Standard generators for $S_6(7)$ are a pair (x, y) such that x is in class $7A/B$, y has order 9, xy has order 57 and xy^2 has order 12.*

Checking these generators requires a way of showing that x is a transvection. A transvection in G has centralizer $7^{1+4} : 2S_6(7)$ of index 58824 in G . This index is small enough that it is feasible to perform a random search of words to find elements commuting with x . We found:

$$C_G(x) = \langle x^6 y^2 x y^7 x y^2, x^6 y x y^8 x y \rangle \quad (6.3.12)$$

By (6.3.10) above, any element of order 7 commuting with an element of order divisible by 5 is a transvection. We have:

$$\begin{aligned} S_6(7) \approx \langle \langle x, y, (z) \mid \mathbf{std}, o(z) = 10, o([x, z]) = 1; \\ z = (x^6 y^2 x y^7 x y^2)^3 (x^6 y x y^8 x y)^5 \rangle \rangle \end{aligned} \quad (6.3.13)$$

6.3.3 Symplectic group $S_{10}(3)$

Let $G = S_{10}(3)$. The group G contains two conjugacy classes of transvections, called $3A$ and $3B$, and these fuse to a single class $3A/B$ in the automorphism group of G . We will generate the group G with a transvection and another element. We first show that the class of transvections is tame.

Lemma 6.27 *We have:*

$$T(3A/B) \supseteq \{63, 117, 120, 123, 126, 180, 234, 240, 246, 252\} \quad (6.3.14)$$

Proof. We will make use of the following fact. Let p be a prime. If the Sylow- p subgroups of G are cyclic of order p , then any element of G with order divisible by p is contained in a conjugate of $H_p = C_G(t_p)$, where t_p is an arbitrary element of order p in G .

(i) $120, 180 \in T(3A/B)$

Let H be a Sylow 5-subgroup of G ; it is elementary abelian of order 25. By calculating in H , we can see that G has exactly 2 conjugacy classes $5A$ and $5B$ containing elements of order 5.

- The centralizer $C_G(5A)$ has shape $5 \times 2.S_6(3)$. We have $24, 36 \in T_{\text{Sp}_6(3)}(3A/B)$, where $3A/B$ is the class of transvections in $\text{Sp}_6(3)$.
- The centralizer $C_G(5B)$ has order 86400. It contains elements of order 5×24 , all of which power to transvections. It does not contain any elements of order 5×36 .

Hence any element with order 5×24 or 5×36 powers to a transvection in G .

(ii) $63 \in T(3A/B)$

The Sylow 7-subgroups of G are cyclic, and the subgroup H_7 has shape $28 \circ \text{Sp}_4(3)$. The class $3A/B$ of $\text{Sp}_4(3)$ corresponds to the class of transvections in G . Because $9 \in T_{\text{Sp}_4(3)}(3A/B)$, any element of order 7×9 must be in H_7 and must power up to a transvection.

(iii) $117 \in T(3A/B)$

The Sylow 13-subgroups of G are cyclic, and the subgroup H_{13} has shape $13 \times \text{Sp}_4(3)$. By essentially the same argument as for $p = 7$, any element of order 13×9 must power up to a transvection.

(iv) $123 \in T(3A/B)$

The Sylow 41-subgroups of G are cyclic, and the subgroup H_{41} has structure $41 \times \text{Sp}_2(3)$, and all elements of order 3 therein are transvections. Thus any element of G of order 41×3 must power up to a transvection.

To get the remaining orders in (6.3.14), we remark that if $n \in T(3A/B)$ and G contains elements of order kn for an integer $k \geq 1$, then $kn \in T(3A/B)$. All element orders in (6.3.14) have been observed in G . ■

Remark Experiments suggest $T(3A/B) \supseteq \{27, 48\}$, but the Sylow-2 and Sylow-3 subgroups were too large to deal with.

The group G can be $(3A/B, 11)$ -generated, and there is a unique conjugacy class of elements of order 11. This can be seen by observing that the Sylow-11 subgroup of G is cyclic of order 121, and that if g has order 11, then g and g^2 are conjugate. We decided to $(3A/B, 11, 121)$ -generate the group. After a search of approximately $1\frac{1}{2}$ minutes, we found the 244 orbits of $C_G(11) = C_{121}$ on the conjugacy class $3A$, 22 of which corresponded to $(3A, 11, 121)$ -pairs.

Definition-Theorem 6.28 Standard generators of $S_{10}(3)$ are a pair (x, y) such that x is in class $3A/B$, y has order 11, xy has order 121 and xy^2 has order 11.

Remark The standard finder takes longer to terminate on average than for most of the other groups. Elements $h \in 11A$ have small centralizers, but elements $h' \in G \setminus 11A$ which satisfy $\langle g, h' \rangle = G$ and $|C_G(h')| > |C_G(h)|$ lie in conjugacy classes which are difficult to distinguish.

For a checker, we need to show that x is a transvection, which we can do by finding an element of order 41 which commutes with it (this is not too hard because $C_G(x)$ has

Group	Average cost ^a
L ₃ (13)	45
L ₄ (4)	56
L ₄ (5)	53
L ₅ (3)	67
U ₃ (9)	33
U ₃ (13)	81
U ₃ (16)	46
U ₄ (4)	81
U ₄ (5)	36
U ₅ (3)	37
U ₅ (4)	100
U ₆ (3)	110
U ₇ (2)	31
U ₈ (2)	58
S ₄ (8)	28
S ₄ (9)	43
S ₆ (5)	160
S ₆ (7)	373
S ₈ (3)	66
S ₁₀ (3)	249
O ₁₀ ⁺ (2)	79

^a1000 trials

Table 6.4: Average cost of running black box algorithms

index 29524 in G). We have:

$$S_{10}(3) \approx \langle \langle x, y, (z) \mid \mathbf{std}, o(z) = 41, o([x, z]) = 1; z = xy^3xy^8xy^3 \rangle \rangle \quad (6.3.15)$$

6.4 Cost of the black box algorithms

We measured the cost of running each black box algorithms, where the cost is defined to be the number of times the algorithm asked for a random element of the group. The results averaged over 1000 trials are given in Table 6.4.

6.5 Checkers for the remaining groups

We will need checkers for all groups where we produce representations. This is because we will use the checkers to verify that the representations we produce are correct. There are a number of groups which have standard generators defined in the Web Atlas but for which we have not yet given a semi-presentation. On the accompanying CD-ROM (see Appendix C) we provide checkers for all these groups except ${}^3D_4(3)$ (where we were not able to find any representations either, so there is less need for a checker). These checkers were made in a similar way to those described in this chapter.

Part II

Characteristic zero representations

Chapter 7

Known constructions

In this chapter, we describe some known constructions for representations of simple groups which we exploited when producing our database of representations. In particular, we describe:

- The representations of A_n
- The representations of $L_2(q)$
- The Weil representations of $S_{2n}(q)$ with q odd

We also make use of some known constructions for certain representations of the sporadic groups.

7.1 Representations of A_n

We will approach the representations of the alternating groups by restricting representations of the symmetric groups (and decomposing where necessary). A great deal is known about the representations of S_n . In this section, we will provide a summary of the facts we need, as well as a construction of the irreducible representations of S_n . An exposition of this material can be found in [43].

Definition 7.1 A partition of n is a non-increasing tuple $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_r)$ of positive integers which sums to n . We write $\lambda \vdash n$. Such a tuple defines subsets $\Omega_i \subseteq \Omega$ for $1 \leq i \leq r$ as follows:

$$\Omega_i = \{\mu_i + 1, \mu_i + 2, \dots, \mu_i + \lambda_i\} \quad (7.1.1)$$

where $\mu_i = \sum_{j=0}^{i-1} \lambda_j$. These subsets define the partition of Ω determined by λ .

Definition 7.2 Let $\lambda = (\lambda_1, \dots, \lambda_r)$ be a partition of n .

1. The Young subgroup S_λ of λ is the stabilizer of the partition of Ω determined by λ . It is isomorphic to $S_{\lambda_1} \times S_{\lambda_2} \times \dots \times S_{\lambda_r}$.
2. A Young (or Ferrers) diagram of shape λ is an array of n cells arranged into r rows with λ_i cells in the i th row. Cells are lined up into columns and each row is left-justified.
3. Given a cell x in a Young diagram, its hook length $\ell(x)$ is the length of the longest Γ -shaped hook whose corner is in the cell.
4. A Young tableau of shape λ is a Ferrers diagram where each cell contains an element of Ω and each element of Ω is contained in at least one cell. A Young tableau is said to be standard if the entries increase along each row and down each column.
5. Young diagrams and tableaux can be transposed by swapping rows and columns. The dual partition λ^\top is that partition of n which corresponds to the transpose of the Young diagram of λ .
6. A Young tabloid of shape λ is an equivalence class of Young tableaux, where two tableaux t_1, t_2 are equivalent if each row of t_1 is a reordering of the corresponding row of t_2 . The equivalence class containing a Young tableau t will be denoted $\{t\}$.

There is an action of S_n on the set of Young tableaux of shape λ , and it is equivalent to the regular action of S_n on itself. In this action, the Young subgroup S_λ is the sta-

bilizer of a particular Young tabloid, and we get an induced action of S_n on the set of Young tabloids. This determines a CS_n -module isomorphic to the trivial CS_λ -module induced up to S_n . We will call this module M_λ .

Definition 7.3 *Let λ be a partition of n .*

1. *Let t be a Young tableau of shape λ . Then the polytabloid e_t associated to t is the following element of M_λ :*

$$e_t = \sum_{\pi} \text{sgn}(\pi) \{t\}^\pi \quad (7.1.2)$$

where the summation is over permutations $\pi \in S_n$ which preserve the columns of the tableau t .

2. *The Specht module V_λ is the submodule of M_λ spanned by the polytabloids e_t , where t ranges over the Young tableaux of shape λ .*

Theorem 7.4 *The irreducible CS_n -modules are the Specht modules V_λ where λ ranges over partitions of n . A basis for this module is given by $\{e_t\}$, where t ranges over standard Young tableaux. The size of this module (and hence the number of standard Young tableaux) is given by the hook length formula:*

$$\rho(1) = \frac{n!}{\prod_{x \in \lambda} \ell(x)} \quad (7.1.3)$$

The basis of standard polytabloids for the Specht modules gives rise to the *Young natural representation* for S_n . There are efficient ways of computing it [43].

Lemma 7.5 *Let λ be a partition of n , with ρ the corresponding irreducible representation of S_n .*

1. *If $\lambda = \lambda^\top$, then $\rho|_{A_n}$ splits into two complex conjugate irreducible representations.*
2. *If $\lambda \neq \lambda^\top$, then $\rho|_{A_n}$ is irreducible. Moreover, if σ is the irreducible representation corresponding to λ^\top , then $\rho|_{A_n} = \sigma|_{A_n}$.*

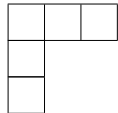
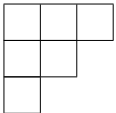
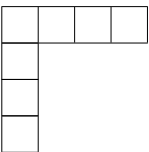
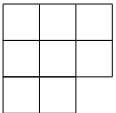
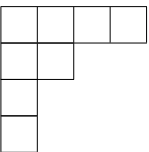
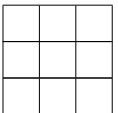
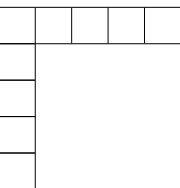
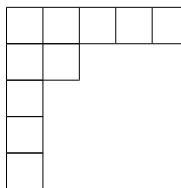
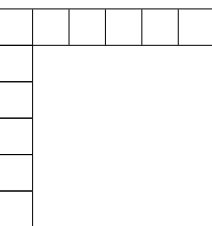
				
$n = 5$ $d = 3$	$n = 6$ $d = 8$	$n = 7$ $d = 10$	$n = 8$ $d = 21$	$n = 8$ $d = 45$
				
$n = 9$ $d = 21$	$n = 9$ $d = 35$	$n = 10$ $d = 224$	$n = 11$ $d = 126$	

Table 7.1: Young diagrams for representations of S_n splitting in A_n

Using the hook length formula (7.1.3) we can determine which partitions give rise to representations of dimension ≤ 250 , and we can determine those which split in A_n . Using this data, we construct the Young natural representation on a set of generators for A_n , splitting by hand where necessary.

The representations of A_n with genera in \mathcal{L}_0 which are obtained by splitting representations of S_n are given in Table 7.1.

7.1.1 The Young system

We have produced a set of GAP programs called Young which can calculate the Young representations for the symmetric groups. (We could also have used the permutation representation methods of Chapter 8 for this purpose.) The inputs to the main program are a partition and a set of permutations.

Here we give a simple example of its use. The group A_7 has a unique 15-dimensional representation. We can find out which partitions of 7 correspond to this using our implementation of the hook length formula (7.1.3)

```
gap> P := Partitions(7);;
gap> Filtered(P, x->HookLengthFormula(x) = 15);
[ [ 3, 1, 1, 1, 1 ], [ 5, 1, 1 ] ]
```

So the partitions are $[3, 1^4]$ and $[5, 1^2]$. These partitions are dual to each other, so the corresponding representations of A_7 are equivalent:

```
gap> DualPartition([3,1,1,1,1]);
[ 5, 1, 1 ]
```

We can take the standard generators of A_7 to be $(1, 2, 3)$ and $(3, 4, 5, 6, 7)$, and hence calculate the appropriate representation:

```
gap> M := YoungRepresentation([3,1,1,1,1], [(1,2,3), (3,4,5,6,7)]);
[ [ [ 1, 0, 0, 0, 0, -1, 1, -1, 1, 0, 0, 0, 0, 0, 0 ],
    [ 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0 ],
    [ 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0 ],
    [ 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0 ],
    [ 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0 ],
    [ 0, 1, 0, 0, 0, -1, 0, 0, 0, 1, -1, 1, 0, 0 ],
    [ 0, 0, 1, 0, 0, 0, -1, 0, 0, 1, 0, 0, -1, 1 ],
    [ 0, 0, 0, 1, 0, 0, 0, -1, 0, 0, 1, 0, -1, 1 ],
    [ 0, 0, 0, 0, 1, 0, 0, 0, -1, 0, 0, 1, 0, -1 ],
    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0 ],
    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0 ],
    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0 ],
    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0 ],
    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 ] ],
  [ [ 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
    [ 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
    [ 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
    [ 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
    [ 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0 ],
    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0 ],
    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0 ],
    [ 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0 ],
    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0 ],
    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 ],
    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ] ] ]
```

7.2 Representations of $L_2(q)$

The representations of $L_2(q)$ are given in Table 7.2. Here we explain briefly how to construct each representation.

q	Dimension	Number of representations	Indicator
$q = 2m$	$q - 1$	m	+
	q	1	+
	$q + 1$	$m - 1$	+
$q = 4m + 1$	$(q + 1)/2$	2	+
	$q - 1$	m	+
	q	1	+
	$q + 1$	$m - 1$	+
$q = 4m + 3$	$(q - 1)/2$	2	o
	$q - 1$	m	+
	q	1	+
	$q + 1$	m	+

Table 7.2: Representations of $L_2(q)$

- q -dimensional representations: this representation is the unique faithful constituent of the permutation representation of G on $q + 1$ points, and is easy to find.
- $(q + 1)$ -dimensional representations: these can be found by inducing 1-dimensional representations of a Borel subgroup. These representations are therefore monomial.
- $(q + 1)/2$ -dimensional representations: these can be found by decomposing the permutation of G on $2(q + 1)$ points. The representation decomposes as $1 + [(q + 1)/2]_a + [(q + 1)/2]_b + q$, and we can use the Split-P system in Chapter 8 to do this.
- $(q - 1)/2$ -dimensional representations. These can be found by thinking of $L_2(q)$ as $\mathrm{PSp}_2(q)$ and constructing the Weil representation (see section 7.3).
- $(q - 1)$ -dimensional representations. These are the cuspidal representations, which will be described in the following section.

7.2.1 The cuspidal representations of $L_2(q)$

The construction of the $(q - 1)$ -dimensional representations of $L_2(q)$ is more complicated than the others. We will sketch the construction here, following Piatetski-Shapiro [39], although the construction is described elsewhere [38, 52].

Let $K = \text{GF}(q)$ and let $L = \text{GF}(q^2)$ be the (unique) quadratic extension of K . We fix a non-trivial character Ψ of the additive group K (an elementary abelian p -group). Let V be the $(q - 1)$ -dimensional vector space of functions $f : K^* \rightarrow \mathbb{C}$. We give V a $\text{CSL}_2(q)$ -module structure as follows. Let ν be an irreducible character of L^* . Let

$$g = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \quad (7.2.1)$$

be an element of $\text{SL}_2(q)$. Then for $f \in V$ and $x \in K^*$, we set:

$$(g.f)(x) = \nu(\delta) \psi\left(\frac{\beta x}{\delta}\right) f\left(\frac{\alpha x}{\delta}\right) \quad (7.2.2)$$

if $\gamma = 0$; and

$$(g.f)(x) = \frac{1}{q} \sum_{y \in K^*} \psi\left(\frac{\alpha y + \delta x}{\gamma}\right) \sum_{u \in L^*, u^{q+1} = y/x} \psi\left(-\frac{x(u + u^q)}{\gamma}\right) \nu(u) f(y) \quad (7.2.3)$$

if $\gamma \neq 0$. It turns out that this makes V a $\text{CSL}_2(q)$ -module. Since we are interested in representations of $L_2(q)$, we must make sure that all scalar matrices act trivially. If g is a scalar matrix, then (7.2.2) implies:

$$(g.f)(x) = \nu(\delta) f(x) \quad (7.2.4)$$

Hence we must ensure that $\nu(-1) = 1$.

Unfortunately, while it is very easy to calculate the image of any particular element of $L_2(q)$ in this representation, the matrices that we get are not good to compute with. Each matrix entry involves $q - 1$ terms, and these terms are typically the products of $(q - 1)$ th and $(q + 1)$ th roots of unity. Thus multiplying two matrices in this representation is at least an $\mathcal{O}(q^5)$ operation, and depending on how the terms are represented on a computer (e.g. if sums of n th roots of unity are stored as lists of size n), it could potentially be an $\mathcal{O}(q^7)$ operation.

7.3 Weil representations of symplectic groups

Let $G = \mathrm{Sp}_{2n}(q)$ with $q = p^r$ odd. We can construct a q^n -dimensional representation called the *Weil representation* as follows. (We follow the discussion in Szechtman [51].)

Let $K = \mathrm{GF}(q)$ and let V be a $2n$ -dimensional K -vector space equipped with a non-degenerate symplectic form $\langle \cdot, \cdot \rangle$. Then we can define a group H to be the set:

$$H = \{(c, w) : c \in K, w \in V\} \quad (7.3.1)$$

with multiplication given by:

$$(c, w) \cdot (c', w') = (c + c' + \langle w, w' \rangle, w + w') \quad (7.3.2)$$

Then H is a group of order pq^2n .

We can decompose $V = M \oplus N$ into a direct sum of two totally isotropic subspaces M, N , each having dimension n . Then the set:

$$A = \{(c, w) : c \in K, w \in M\} \quad (7.3.3)$$

is an abelian subspace. By inducing up a linear character of A to H , we get a q^n -

dimensional representation of H . The group G acts by conjugation on H :

$$(c, w)^g = (c, gw) \quad (7.3.4)$$

and this conjugation action gives rise to a q^n -dimensional representation of G , known as a *Weil representation*.

7.3.1 The method

For elements $g \in G$, we find the effect of the action of g on a basis of V , a symplectic space. We rewrite this action of g in terms of the corresponding extraspecial group in a q^n -dimensional representation. We then find (using standard-basis methods [36]) a matrix $\theta(g)$ which performs the corresponding conjugation. The matrix $\theta(g)$ is an element of $G^0 = q^{1+2n} : \mathrm{GSp}_{2n}(q)$, the automorphism group of the extraspecial group q^{1+2n} . We find enough of these elements to generate the whole group. We then search for the subgroup $G = \mathrm{Sp}_{2n}(q)$ in G^0 . We can do this by observing that z , the central involution of G , is not centralized by the extraspecial group in G^0 .

The q^n -dimensional representation then splits into 2 parts. We can perform the splitting by considering the action of the group on the -1 and $+1$ eigenspaces of the central involution z .

7.4 Other known constructions

Certain representations for groups have been constructed already. For the representations listed in Table 7.3, we use these constructions (possibly changing a basis).

Group	Dimension	Indicator	Source
$U_5(2)$	10	—	ATLAS
$G_2(3)$	14	+	ATLAS
${}^2F_4(2)'$	26	○	ATLAS (restricted from $2.F_4(2)$)
M_{24}	45	○	[32, 42]
Fi_{22}	78	+	ATLAS
$Sz(32)$	124	○	John Bray
HN	133	+	[6, 35]
Th	248	+	[48]

Table 7.3: Representations from other sources

Chapter 8

Split-P: a GAP system for splitting permutation modules

In this chapter, we describe our Split-P system, which is a set of GAP programs that can be used to decompose a permutation module into its constituents.

We begin by giving an account of the theory of intersection matrices, and show how it applies to the situation of decomposing permutation modules. We then describe *rational reconstruction*: a technique which allows us to perform most of our calculations in finite fields for faster computation. Finally we give an outline of the implementation of Split-P, and show an example of its use.

The general theory up to and including Algorithm 8.8 is well-known and was used by Rogers [42] to construct several representations of sporadic groups.

8.1 Notation

Let G be a finite group acting transitively on a set Ω of size n . Then G has a permutation representation:

$$\theta : G \rightarrow S_n \tag{8.1.1}$$

We will write α^g for the result of the action of g on a point $\alpha \in \Omega$. The *permutation module* $\overline{\Omega}$ is an n -dimensional vector space over \mathbb{C} with basis:

$$\{\overline{\alpha} : \alpha \in \Omega\} \quad (8.1.2)$$

where the G -action is defined by setting:

$$\overline{\alpha}^g = \overline{\alpha^g} \quad (8.1.3)$$

and extending linearly to the whole of $\overline{\Omega}$. Let χ be the permutation character.

There is an action of G on $\Omega \times \Omega$ given by:

$$(\alpha, \beta)^g = (\alpha^g, \beta^g) \quad (8.1.4)$$

For $n \geq 2$, this action is not transitive, because the set

$$\Lambda_0 = \{(\alpha, \alpha) : \alpha \in \Omega\} \quad (8.1.5)$$

is an orbit. Let the other orbits be labelled $\Lambda_1, \Lambda_2, \dots, \Lambda_d$. The complete set of orbits is denoted \mathcal{O} .

For any $\alpha \in \Omega$, $0 \leq i \leq d$, we set:

$$\Lambda_i(\alpha) = \{\beta \in \Omega : (\alpha, \beta) \in \Lambda_i\} \quad (8.1.6)$$

The sets $\Lambda_i(\alpha)$ for fixed α are the orbits of $\text{Stab}_G(\alpha)$ on Ω .

Observe that for any orbit Λ of G on $\Omega \times \Omega$, the set Λ^* defined by:

$$\Lambda^* = \{(\beta, \alpha) : (\alpha, \beta) \in \Lambda\} \quad (8.1.7)$$

is also an orbit. The map

$$(\)^* : \mathcal{O} \rightarrow \mathcal{O} \quad (8.1.8)$$

$$\Lambda \mapsto \Lambda^* \quad (8.1.9)$$

is known as the *pairing operator* on \mathcal{O} . It is either a trivial map or an involution. The fixed points of the pairing operator are known as *self-paired orbits*. The pairing operator induces a permutation $*$: $j \mapsto j^*$ of the set $\{0, 1, \dots, d\}$ by the relation:

$$\Lambda_{j^*} = (\Lambda_j)^* \quad (8.1.10)$$

for $0 \leq i \leq d$. This permutation extends to a linear map on \mathbb{C}^{d+1} . For standard basis vectors e_i ($0 \leq i \leq d$) we define:

$$(e_i)^* = e_{i^*} \quad (8.1.11)$$

and extend linearly to the whole of \mathbb{C}^{d+1} .

For convenience, we impose a total order on Ω as follows. Choose an arbitrary element $\omega_0 \in \Omega$, and choose an arbitrary total order \leq_i on $\Lambda_i(\omega_0)$ for each $0 \leq i \leq d$. For $\alpha \in \Lambda_i(\omega_0)$ and $\beta \in \Lambda_j(\omega_0)$ we set:

$$\alpha \leq \beta \Leftrightarrow \begin{cases} i < j & \text{or} \\ i = j \text{ and } \alpha \leq_i \beta \end{cases} \quad (8.1.12)$$

The effect of this total ordering is that we may think of Ω as the set $\{1, 2, \dots, n\}$ where each orbit of $\text{Stab}_G(1)$ is a set of consecutive integers. When we give linear maps $\overline{\Omega} \rightarrow \overline{\Omega}$, they will be given as matrices with respect to the standard basis:

$$\{\bar{\alpha} : \alpha \in \Omega\} \quad (8.1.13)$$

written in the order given by \leq .

Define ω_i for $1 \leq i \leq d$ to be the \leq -minimal element of $\Lambda_i(\omega_0)$ (note that this definition also applies for $i = 0$).

8.2 The centralizer algebra \mathcal{A}

The *centralizer algebra* \mathcal{A} of the module $\overline{\Omega}$ is the set of linear maps $\overline{\Omega} \rightarrow \overline{\Omega}$ which commute with the action of G . By Schur's Lemma, the structure of \mathcal{A} is determined by the decomposition of $\overline{\Omega}$ into irreducible constituents. Suppose $\overline{\Omega}$ decomposes as:

$$\overline{\Omega} = \bigoplus_{i=1}^r m_i W_i \quad (8.2.1)$$

where the summands W_i ($1 \leq i \leq r$) are distinct irreducible $\mathbb{C}G$ -modules and the multiplicities m_i are positive integers. Then the centralizer algebra has structure:

$$\mathcal{A} \cong \bigoplus_{i=1}^r M_{m_i}(\mathbb{C}) \quad (8.2.2)$$

where $M_j(\mathbb{C})$ denotes the algebra of $j \times j$ complex matrices.

This correspondence can be exploited in the other direction. Vector subspaces which correspond to the summands M_{m_i} in (8.2.2) can be used to find the $\mathbb{C}G$ -submodules $m_i W_i$ of $\overline{\Omega}$. In particular, if $m_i = 1$, the matrix algebra M_{m_i} is just \mathbb{C} , and so \mathcal{A} has eigenspaces. An eigenvector v of \mathcal{A} must lie in an irreducible $\mathbb{C}G$ -submodule of $\overline{\Omega}$, and hence we can reconstruct the submodule by letting G act on v (we will see how to do this in section 8.7).

8.3 Adjacency matrices

Let $A : \overline{\Omega} \rightarrow \overline{\Omega}$ be an element of the centralizer algebra. Recall that we can think of linear maps $\overline{\Omega} \rightarrow \overline{\Omega}$ as $n \times n$ matrices where the rows and columns are labelled by

elements of Ω (and written in the order determined by \leq). Thus A is determined by the complex numbers $A_{\alpha\beta}$ for $\alpha, \beta \in \Omega$, and the action of the matrix A on an element $v \in \overline{\Omega}$ is given by:

$$(vA)_\beta = \sum_{\alpha \in \Omega} v_\alpha A_{\alpha\beta}, \quad \beta \in \Omega \quad (8.3.1)$$

The effect of conjugating such a matrix by an element of G is:

$$(gAg^{-1})_{\alpha\beta} = A_{\alpha^g\beta^g}, \quad \alpha, \beta \in \Omega \quad (8.3.2)$$

and because A commutes with the action of G , its (α, β) components must be constant for all (α, β) pairs in each orbit Λ . This observation leads us to the following definition.

Definition 8.1 *The i -th adjacency matrix A_i is the $n \times n$ matrix given by:*

$$(A_i)_{\alpha\beta} = \begin{cases} 1 & \text{if } (\alpha, \beta) \in \Lambda_i \\ 0 & \text{otherwise} \end{cases} \quad (8.3.3)$$

for $\alpha, \beta \in \Omega$.

Theorem 8.2 *The adjacency matrices A_i , $0 \leq i \leq d$ form a (vector space) basis of the centralizer algebra \mathcal{A} .*

Proof. The adjacency matrices span \mathcal{A} by our observation above, and they are linearly independent, because the (ω_0, ω_i) co-ordinate of A_j is 1 if and only if $i = j$. ■

Corollary 8.3 *We have:*

$$\langle \chi, \chi \rangle = \sum_{i=1}^r m_i^2 = d + 1 \quad (8.3.4)$$

Proof. Take the character of (8.2.1), and compute the dimension of \mathcal{A} from (8.2.2). ■

8.4 The left-regular representation of \mathcal{A}

Most of the time, $n \times n$ matrices are too large to handle easily. We would like a way to transfer our calculations to a smaller representation of \mathcal{A} . This is provided by the *left-regular representation*.

The structure constants of \mathcal{A} with respect to the basis A_0, A_1, \dots, A_d are given by the numbers p_{ij}^k ($0 \leq i, j, k \leq d$) in the following equation:

$$A_i A_j = \sum_{k=0}^d p_{ij}^k A_k, \quad 0 \leq i, j \leq d \quad (8.4.1)$$

Let B_i be the matrix with co-ordinates given by:

$$(B_i)_{jk} = p_{ij}^k, \quad 0 \leq j, k \leq d \quad (8.4.2)$$

Then if e_i is the i th standard basis vector in \mathbb{C}^{d+1} ($0 \leq i \leq d$), then

$$(e_j B_i)_k = \sum_{\ell=0}^d (e_j)_\ell (B_i)_{\ell k} \quad (8.4.3)$$

$$= \sum_{\ell=0}^d \delta_{j\ell} (B_i)_{\ell k} \quad (8.4.4)$$

$$= (B_i)_{jk} \quad (8.4.5)$$

$$= p_{ij}^k \quad (8.4.6)$$

whence:

$$e_j B_i = \sum_{k=0}^d p_{ij}^k e_k \quad (8.4.7)$$

(compare to equation (8.4.1)). So the homomorphism $\mathcal{A} \rightarrow M_{d+1}(\mathbb{C})$ induced by $A_i \mapsto B_i$ is the left-regular representation of \mathcal{A} . We will denote the image of this map by \mathcal{B} .

Now, the structure constants p_{ij}^k have a combinatorial description:

$$(A_i A_j)_{\alpha\beta} = \sum_{\gamma \in \Omega} (A_i)_{\alpha\gamma} (A_j)_{\gamma\beta} \quad (8.4.8)$$

$$= |\{\gamma \in \Omega : (\alpha, \gamma) \in \Lambda_i, (\gamma, \beta) \in \Lambda_j\}| \quad (8.4.9)$$

$$= |\Lambda_i(\alpha) \cap \Lambda_j^*(\beta)| \quad (8.4.10)$$

Thus:

$$p_{ij}^k = |\Lambda_i(\omega_0) \cap \Lambda_j^*(\omega_k)| \quad (8.4.11)$$

These numbers are quite easy to compute if n is not too large, and hence by equation (8.4.2) we can calculate matrices for the left regular representation of \mathcal{A} without needing to compute the adjacency matrices.

8.5 Eigenvectors of adjacency matrices

Recall from section 8.2 that we are interested in eigenvectors of \mathcal{A} . Since the adjacency matrices A_i span \mathcal{A} , it is sufficient to consider simultaneous eigenvectors of the A_i . In this section, we will see how the eigenvectors of $\{A_i\}$ are related to those of $\{B_i\}$. This connection is useful, because the matrices B_i are considerably smaller and easier to calculate with.

The key ingredient is the *stretching operator*, defined by:

$$\begin{aligned} \widehat{(\cdot)} : \mathbb{C}^{d+1} &\rightarrow \mathbb{C}^n \\ v &\mapsto \hat{v} \end{aligned} \quad (8.5.1)$$

where for $v = (v_0, v_1 \dots v_d)$ we have:

$$\hat{v} = (\underbrace{v_0, \dots, v_0}_{|\Lambda_0(\omega_0)| \text{ times}}, \underbrace{v_1, \dots, v_1}_{|\Lambda_1(\omega_0)| \text{ times}}, \dots, \underbrace{v_d, \dots, v_d}_{|\Lambda_d(\omega_0)| \text{ times}}) \quad (8.5.2)$$

Here we use our particular ordering for the basis vectors given in (8.1.12).

We will also need the following preliminary lemma.

Lemma 8.4 For $0 \leq i, j, k \leq d$ we have:

$$p_{ij}^k = p_{j^*i^*}^{k^*} \quad (8.5.3)$$

Proof. Observe that $A_{i^*} = A_i^\top$. By taking transposes of equation (8.4.1) we get

$$A_{j^*} A_{i^*} = \sum_{k=0}^d p_{ij}^k A_{k^*} \quad (8.5.4)$$

By relabelling coefficients in equation (8.4.1) we have:

$$A_{j^*} A_{i^*} = \sum_{k=0}^d p_{j^*i^*}^{k^*} A_{k^*} \quad (8.5.5)$$

The result follows by comparing coefficients of A_{k^*} . ■

Theorem 8.5 Let v be an eigenvector of B_i with eigenvalue λ for some $0 \leq i \leq d$. Then \hat{v}^* is an eigenvector of A_{i^*} with eigenvalue λ .

Proof. Let $\beta \in \Omega$ be arbitrary, and suppose $(\omega_0, \beta) \in \Lambda_k$. Then

$$(\hat{v}^* A_{i^*})_\beta = \sum_{\alpha \in \Omega} (\hat{v}^*)_\alpha (A_{i^*})_{\alpha\beta} \quad (8.5.6)$$

$$= \sum_{\substack{\alpha \in \Omega \\ (\alpha, \beta) \in \Lambda_{i^*}}} (\hat{v}^*)_\alpha \quad (8.5.7)$$

$$= \sum_{j=0}^d \sum_{\substack{\alpha \in \Lambda_j(\omega_0) \\ (\alpha, \beta) \in \Lambda_{i^*}}} (v^*)_j \quad (8.5.8)$$

$$= \sum_{j=0}^d |\Lambda_j(\omega_0) \cap \Lambda_{i^*}(\beta)| v_{j^*} \quad (8.5.9)$$

$$= \sum_{j=0}^d p_{ji^*}^k v_{j^*} \quad (8.5.10)$$

$$= \sum_{j=0}^d p_{ij^*}^{k^*} v_{j^*} \quad (\text{by Lemma 8.4}) \quad (8.5.11)$$

$$= \sum_{j=0}^d p_{ij}^{k^*} v_j \quad (8.5.12)$$

$$= \sum_{j=0}^d (B_i)_{jk^*} v_j \quad (8.5.13)$$

$$= \lambda v_{k^*} \quad (8.5.14)$$

$$= (\lambda \widehat{v^*})_\beta \quad (8.5.15)$$

So $\widehat{v^*} A_{i^*} = \lambda \widehat{v^*}$, as required. ■

In practice we do not worry about the pairing operator $(\)^*$, as it is often sufficient to know that we have a vector \widehat{v} which is the eigenvector for some element of \mathcal{A} .

Corollary 8.6 *If v is an eigenvector of B_i for all $0 \leq i \leq d$ then \widehat{v} is an eigenvector of A_i for all $0 \leq i \leq d$.*

8.6 Finding eigenvectors of elements of \mathcal{B}

Theorem 8.5 reduces the problem of finding eigenvectors of adjacency matrices to that of finding eigenvectors of elements B of \mathcal{B} ; that is, $(d+1) \times (d+1)$ matrices.

In some cases, d is sufficiently small that we can use relatively naïve procedures for finding eigenvectors. We find and factorise the characteristic polynomial $p(x)$ of B and then find nullspaces of matrices $B - \lambda I$ for roots λ of $p(x)$.

Linear and quadratic factors of $p(x)$ occurring with multiplicity 1 are easy to deal with. We are not currently able to deal with the irreducible factors of higher order.

8.7 Spinning up

We now need to describe a technique which originated with Parker's Meataxe [36]. Let G a finite group generated by g_1, \dots, g_m . Let k be a field, and V a finite-dimensional kG -module. Let $\{w_1, w_2, \dots, w_k\}$ be a non-empty linearly independent subset of V and let W denote the kG -submodule of V generated by these vectors.

Algorithm 8.7 (Classical spinning up) *To produce a basis S of W containing the vectors w_1, \dots, w_k :*

1. Let $S = (w_1, \dots, w_k)$ be an expandable list of vectors.
2. For each w in L and each g in the set of generators
 - (a) Construct $v = wg$
 - (b) If v is not in the k -span of L , add v to L
 - (c) If $|S| = \dim W$, stop: S is the required basis.
3. The space kL is now G -invariant, and S is the required basis.

The particular case we will be most interested in is when we have a single vector w_1 in an irreducible kG -submodule of V . In this cases, S is a basis for W .

Step 2b can be quite costly computationally. The implementation usually requires a basis in echelon form to be maintained alongside the basis S . As we will see later in section 8.10, if $k = \mathbb{Q}$ and we know $\dim W$, we can approximate this step by reducing modulo p for some prime p .

8.8 Splitting in \mathbb{C}

We have now given enough theory to present the basic implementation of Split-P.

Algorithm 8.8 (General Split-P) *To split the permutation module $\overline{\Omega}$:*

1. *Calculate the suborbits $\Lambda_i(1)$ and conjugate the generators of G by a suitable element of S_n to make the suborbits consist of consecutive integers (so that the order \leq from (8.1.12) is just the usual order on \mathbb{Z}).*
2. *Calculate the matrices B_i using equations (8.4.2) and (8.4.11).*
3. *Find as many linearly independent simultaneous eigenvectors of the matrices B_i as possible using the techniques of section 8.6. Allow extra vectors to be found by hand if we do not have a basis.*
4. *Expand the eigenvectors using the stretching operator $\widehat{(\)}$. By Theorem 8.5 these will give eigenvectors of the matrices A_{i^*} .*
5. *Given a stretched eigenvector f , spin it up to give a basis f_1, \dots, f_t of a submodule $W \subseteq \overline{\Omega}$. Here we take advantage of the fact that G is a permutation group, so the action of a group element on a vector can be calculated by permuting co-ordinates, which is much faster than matrix multiplication.*
6. *Find matrices for the action of G on the modules by expressing the vectors $f_i g$ as linear combinations of the basis vectors f_1, \dots, f_t for each generator $g \in G$ and $1 \leq i \leq t$. This is essentially a Gaussian elimination problem.*

At step 3 we may require some of the vectors to be found by hand: we do not need all the vectors to be simultaneous eigenvectors, but not all eigenvectors of a particular matrix B_i will be useful. It is always helpful to know what the decomposition of $\overline{\Omega}$ will

look like: for this, we use the character table of G and the permutation character of $\overline{\Omega}$ and take scalar products.

Note that some of the time we will not be interested in all the irreducible submodules of Ω : some of them will be more than 250-dimensional. Thus we also needed some way of abandoning step 5 if the module was going to be too big. This is because we do not know of a definite way of telling in advance which irreducible submodule will be produced by a given eigenvector, although it is possible, for example, to tell from the eigenvectors which modules will occur in complex conjugate pairs.

8.9 Computational problems with the field \mathbb{Q}

We now change track slightly, and explain why Algorithm 8.8 needed to be improved.

A phenomenon which frequently occurs in computational algebra is that of *expression swell*; that is, in exact computations, the numbers and expressions grow dramatically in size as a calculation progresses, even if the final answer does not involve very large numbers. This often causes a problem when performing calculations in \mathbb{Q} . Using the ‘obvious’ algorithms, the cost of adding or multiplying two rational numbers where the numerators and denominators have k digits is $\mathcal{O}(k^2)$,¹ so the cost of a single field operation increases significantly as the numbers grow larger in size. This can cause the entire calculation (which may have a reasonable complexity under the assumption that field operations are $\mathcal{O}(1)$) to become extremely slow and memory-intensive.

Finite fields on the other hand do not suffer from this problem. In **GAP**, finite fields are implemented with a Zech logarithm representation. Non-zero elements $x \in \text{GF}(q)$

¹Asymptotically better behaviour is possible. Adding rational numbers requires the ability to multiply, add and take the gcd of arbitrary integers. Multiplying rational numbers requires multiplying and taking the gcd. Adding integers cannot be done faster than $\mathcal{O}(k)$, but there exist algorithms for gcd which are $\mathcal{O}(k^2 / \log k)$ [12, 49] and an algorithm for multiplying two integers which is $\mathcal{O}(k \log k \log \log k)$ [44].

are represented by an integer $0 \leq i < q - 1$ called the *logarithm* of x . The logarithm i satisfies $x = \omega^i$ (where ω is a fixed primitive element of $\text{GF}(q)$). The zero element is represented by $\omega^{-\infty}$. We precompute a table (the *successor table*) which associates each logarithm i with another logarithm j such that:

$$\omega^j = 1 + \omega^i \quad (8.9.1)$$

For non-zero elements, we have:

$$\omega^i \omega^j = \omega^{i+j} \quad (8.9.2)$$

$$\omega^i + \omega^j = \omega^i (1 + \omega^{j-i}) \quad (8.9.3)$$

Thus all field operations are cheap (and in particular, $\mathcal{O}(1)$) involving only addition and subtraction of (small) integers, taking residues modulo $q - 1$ and looking up values in the successor table.

Gaussian elimination gives a typical example of the difference between the two types of behaviour. Traditional complexity analysis shows that Gaussian elimination is $\mathcal{O}(n^3)$ under the assumption that field operations are $\mathcal{O}(1)$. However, we have seen that this assumption is false in \mathbb{Q} , and the real complexity is much worse than $\mathcal{O}(n^3)$. Table 8.1 shows the time taken by **GAP** to find the inverse of a random $n \times n$ matrix with entries in $[-20, 20] \subset \mathbb{Q}$ and also in several finite fields of prime order. If Gaussian elimination were $\mathcal{O}(n^3)$ for \mathbb{Q} , we would expect the time for $n = 100$ to be roughly 8 times larger than the time for $n = 50$. In fact it is 20 times larger. The timings for the finite fields are consistent with the expected $\mathcal{O}(n^3)$ complexity. For large matrices, the timings are also significantly better in the finite field case. The cost of a field operation is (for our purposes) independent of the size of the finite field once the successor table has been calculated.

n	Time (seconds) to invert matrix with entries in $[-20, 20] \cap \mathbb{Z}$				
		GF(11)	GF(101)	GF(1009)	GF(10007)
30	0.87	0.01	0.01	0.01	0.01
40	3.0	0.01	0.02	0.02	0.02
50	7.6	0.03	0.02	0.02	0.03
60	16	0.04	0.05	0.04	0.04
70	32	0.06	0.07	0.07	0.08
80	58	0.10	0.11	0.10	0.10
90	96	0.14	0.15	0.15	0.15
100	152	0.18	0.20	0.21	0.21

Table 8.1: Cost of Gaussian elimination

Considerations of expression swell apply to storage costs as well as computation times. With large matrices, computing the inverse may cause the computer to run out of memory even though the inverse itself can easily fit into memory.

8.10 Reduction modulo p

The significantly faster performance of **GAP** for finite fields suggests that we should consider whether any of the calculations in Algorithm 8.8 can be performed in a finite field.

The slow steps are steps 5 and 6. The speed of step 5 can be significantly increased by the following simple trick:

Algorithm 8.9 (Fast spin up) *To spin up a vector v to a submodule W of $\overline{\Omega}$:*

1. Choose a prime p .
2. Reduce the vector v modulo p to \bar{v} .
3. Let $S = \{\bar{v}\}$.
4. Spin up the vector \bar{v} . Whenever a new vector $\bar{v}g$ gets added to the modulo p basis, add the equivalent complex vector vg to the set S . (The new complex vector vg is linearly

independent of the rest of S , because otherwise there would be a linear dependence among the modulo p vectors.)

5. *If the size of S is smaller than $\dim(W)$ then choose a different value of p and go back to step 2. Otherwise S is a basis of W .²*

Note that the basis we end up with may be different to that found by Algorithm 8.7 (although for most values of the prime p , the bases will be the same).

8.11 Rational reconstruction

Rational reconstruction uses the observation that finite field operations are much more efficient than rational arithmetic to speed up calculations in \mathbb{Q} . Let C denote a calculation which we wish to perform in \mathbb{Q} . For a well-chosen prime p , we transfer the calculation to an analogous calculation C_p over $\text{GF}(p)$ by reducing modulo p . Because reducing modulo p involves a loss of information, we usually have to do this for several primes. We then merge the results of all the C_p calculations to give an answer for the original calculation C , and check that it is correct. If it is not correct, we try again with more primes. The process of combining the results of the C_p to give a single result for C is called *rational reconstruction*.

There are definite speed gains from dealing with finite field operations rather than rational field operations, and performing the same calculation for many different primes is still usually faster than performing the calculation once for the rationals. Even taking into account the time taken to make modulo p reductions and perform the reconstruction, there is often still an impressive benefit to working this way.

Moreover, working with many finite fields means that there is an easy way to ar-

²If we do not know $\dim(W)$ at this stage, then we assume S is a basis of W . It is certainly a set of linearly independent vectors in W . In the unlikely event that our assumption is wrong and we do not have a complete basis, we will eventually discover this when we attempt to produce our generating matrices for the representation, so any ‘guesses’ made at this stage will be justified by later steps.

range the calculation so that it can be run on parallel processors. We never needed to make use of this facility, but when dealing with very large representations it might be crucial.

8.12 Reconstructing scalars

Before we consider rational reconstruction in its general framework, we will consider the simpler problem of reconstructing a single scalar.

8.12.1 Solution sets

The following definitions are non-standard.

Definition 8.10 *For a rational number x , we can write $x = r/s$ in a unique way with $s > 0$ and $(r, s) = 1$. We define $\text{num}(x) = r$, $\text{denom}(x) = s$.*

Definition 8.11 *For a rational number x , we say a prime p is good for x if p does not divide $\text{denom}(x)$.*

Let x be a rational number and set $r = \text{num}(x)$, $s = \text{denom}(x)$ so that $x = r/s$ in lowest terms. Let P be a finite set of primes good for x . Then x can be considered modulo p for each $p \in P$, because $[s] \in \mathbb{Z}_p$ is an invertible element of \mathbb{Z}_p . Let the residue of x modulo p be denoted m_p , so that we have:

$$x \equiv m_p \pmod{p} \tag{8.12.1}$$

for $p \in P$. We consider (8.12.1) as a set of $|P|$ simultaneous congruences for x . By the Chinese Remainder Theorem, this is equivalent to a single congruence:

$$x \equiv m \pmod{\prod_{p \in P} p} \tag{8.12.2}$$

for some m (which is easy to compute) satisfying $0 \leq m < \prod_{p \in P} p$.

We now consider (8.12.2) as a congruence in an unknown quantity y :

$$y \equiv m \pmod{\prod_{p \in P} p} \quad (8.12.3)$$

Let X_P denote the set of solutions $y \in \mathbb{Q}$ to (8.12.3) for a fixed finite set P of primes good for x . We call X_P the P -solution set for x .

Lemma 8.12 (Properties of solution sets) *Let P, Q be finite sets of primes good for x . Then we have:*

- For all $P, x \in X_P$.
- If $y \neq x$ then $y \notin X_R$ for some set of primes R good for x .
- $X_P \subsetneq X_Q$ if $P \supsetneq Q$.
- X_P is an infinite set.

As a consequence of Lemma 8.12, we will be able to reconstruct x correctly if we take a large enough set of primes good for x .

8.12.2 Reconstruction using the Extended Euclidean Algorithm

The basic technique for reconstructing x is:

1. Take a small set of primes P . Assume that each prime in P is good for x .
2. Calculate the 'best' element x_P in X_P . If we discover a prime $p \in P$ that is not good for x , remove it and try again.
3. Check whether $x_P = x$. If not, enlarge the set P and go back to step 2.

This looks circular, as it appears that we need to know x before we can calculate it. However, x is usually defined by some property that it satisfies, so we can verify instead that x_p satisfies the condition that x is supposed to satisfy.

The rest of this section will be devoted to step 2 of the above procedure. Recall that we are trying to solve the congruence $y \equiv m \pmod{N}$ where $N = \prod_{p \in P} p$.

We apply the Extended Euclidean Algorithm [54] to N and m as follows. Set $n_0 = N$, $n_1 = m$, and perform the Euclid's algorithm for calculating $\gcd(N, m)$:

$$\begin{aligned} n_0 &= q_1 n_1 + n_2 \\ n_1 &= q_2 n_2 + n_3 \\ &\vdots \quad \vdots \quad \vdots \\ n_{r-1} &= q_r n_r + n_{r+1} \\ n_r &= q_{r+1} n_{r+1} + 0 \end{aligned}$$

For each $2 \leq i \leq r+1$ we have $0 < n_r < n_{r-1}$. We can perform backward substitution to get equations of the form:

$$n_i = u_i n_0 + v_i n_1$$

Note that we can find the coefficients u_i, v_i without having to do back-substitution. We set $u_0 = 1, u_1 = 0, v_0 = 0, v_1 = 1$ and using the iteration:

$$\begin{aligned} u_{i+2} &= u_i - q_{i+1} u_{i+1} \\ v_{i+2} &= v_i - q_{i+1} v_{i+1} \end{aligned}$$

Euclid's algorithm	Back-substitution
4199 = 4.900 + 599	900 = 0.N + m
900 = 1.599 + 301	599 = N - 4.m
599 = 1.301 + 298	301 = -N + 5.m
301 = 1.298 + 3	298 = 2.N - 9.m
298 = 99.3 + 1	3 = -3.N + 14.m
3 = 3.1 + 0	1 = 299.N - 1395.m

Table 8.2: Extended Euclidean Algorithm for $N = 4199, m = 900$

Then providing N and v_i are coprime, we get:

$$m \equiv n_i/v_i \pmod{N}$$

and so $n_i/v_i \in X_P$. This method gives several elements of X_P , and our choice for the rational reconstruction of x is that which minimises $\text{ht}(y)$, defined by:

$$\text{ht}(y) = \max(|\text{num}(y)|, |\text{denom}(y)|) \quad (8.12.4)$$

Example 8.13 Let $P = \{13, 17, 19\}$, $N = 13.17.19 = 4199$, $m = 900$. Table 8.2 shows the results of the Extended Euclidean Algorithm. We see the following elements of the solution set X_P :

$$900, -599/4, 301/5, -298/9, 3/14, -1/1395$$

The one with the smallest height is $x_P = 3/14$.

Remark Using a more complicated algorithm, rational reconstruction can be achieved in $\mathcal{O}(k \log^2 k \log \log k)$ time, where k is the number of binary digits in N [56]. The algorithm described above is $\mathcal{O}(k^2)$.

8.13 Reconstruction of quadratic elements

The principles involved in rational reconstruction can apply to infinite fields other than \mathbb{Q} . We extended the method to apply to fields $\mathbb{Q}(\alpha)$ such that α satisfies a quadratic minimal polynomial f . In this case, we can write $\alpha = q_1 + q_2\sqrt{\beta}$ with $q_1, q_2 \in \mathbb{Q}$ and $\beta \in \mathbb{Z}$.

8.13.1 Non-splitting primes

Definition 8.14 *We say a prime p splits f if the polynomial f reduced modulo p is reducible in $\text{GF}(p)$.*

The primes which split f are those in which β is a quadratic residue modulo p . Thus it is easy to determine whether a given prime splits f by using the law of Quadratic Reciprocity.

Let p be a prime which does not split f . Then the field $\text{GF}(p)[X]/(f(X))$ is isomorphic to $\text{GF}(p^2)$. Any element $x \in \mathbb{Q}(\alpha)$ can be written in the form $t/u + \alpha v/w$ where $t, u, v, w \in \mathbb{Z}$. A prime p is *good for x* if p does not divide u or w . In this case, we can define x modulo p to be the element:

$$\bar{x} = \bar{t}\bar{u}^{-1} + X\bar{v}\bar{w}^{-1} \in \text{GF}(p)[X]/(f(X)) \quad (8.13.1)$$

8.13.2 Reconstruction

Let $x = r + \alpha s \in \mathbb{Q}[\alpha]$ be an element which we are interested in reconstructing. Given a set of primes P which are good for x and which do not split f , define x_p to be x modulo p for each $p \in P$. Suppose we can calculate x_p as an element of $\text{GF}(p^2)$. Because p does not split f , each x_p can be written uniquely in the form $r_p + \alpha_p s_p$ for $r_p, s_p \in \text{GF}(p)$, where α_p is one of the roots of f in $\text{GF}(p^2)$. Using rational reconstruction, we can

reconstruct $r, s \in \mathbb{Q}$ separately using the unique decomposition of x_p . Hence we can reconstruct $x \in \mathbb{Q}[\alpha]$.

8.13.3 Problems with irrational reconstruction

One drawback of this technique is that because we have to restrict to fields which do not split f , and each field has prime squared order, we have to start dealing with much larger finite fields than with the rational case. This is a particular problem because GAP does not have support for arbitrarily large finite fields.

The situation only gets worse with field extensions which are of higher degree than quadratic. For this reason, we did not attempt this method with cubic or higher degree field extensions.

8.14 Reconstruction of an action on a submodule

We have found rational reconstruction to be particularly useful in the following situation.

Let G be a permutation group acting on a set Ω and let $\overline{\Omega}$ be the corresponding CG-module. We find a vector $w^{(1)}$ which generates a CG-submodule W of $\overline{\Omega}$. By spinning up, we can find a basis $w^{(1)}, w^{(2)} \dots w^{(n)}$ of W . We wish to find the representation of G corresponding to this basis of W . In other words, for each generator g of G , we wish to find a $n \times n$ matrix g^W which represents the action of g on W with respect to the chosen basis. We have:

$$w^{(i)}g = \sum_{j=1}^n w^{(j)}g_{ij}^W \quad (8.14.1)$$

for $1 \leq i \leq n$, so we wish to find the scalars g_{ij}^W for $1 \leq i, j \leq n$. Thus calculating g^W involves decomposing vectors $v \in W$ into linear combinations of basis elements.

8.14.1 Decomposition into linear combinations of basis elements

Suppose we wish to decompose $v \in W$ into a linear combination:

$$v = \sum_{i=1}^n \lambda_i w^{(i)} \quad (8.14.2)$$

For simplicity, we assume that the basis vectors $w^{(1)}, \dots, w^{(n)}$ are in echelon form. If they are not, then they are converted before our calculations begin and our final decomposition is adjusted accordingly. If the basis vectors are in echelon form, then the decomposition is given by the following iteration:

$$\lambda_i = \frac{v_i^{(i-1)}}{w_i^{(i)}} \quad (8.14.3)$$

$$v^{(i)} = v^{(i-1)} - \lambda_i w^{(i)} \quad (8.14.4)$$

where $1 \leq i \leq n$ and $v^{(0)} = v$. The assumption $v \in W$ implies that $v^{(n)} = 0$.

8.14.2 Complexity analysis

Under the assumption that field operations are $\mathcal{O}(1)$, turning the basis into echelon form is $\mathcal{O}(n^3)$, and calculating each $(\lambda_i, v^{(i)})$ pair in the iteration above is $\mathcal{O}(n)$. Hence decomposing a single vector given a basis in echelon form is $\mathcal{O}(n^2)$, and calculating the matrix g^W for a single generator g is $\mathcal{O}(n^3)$.

Unfortunately, the necessity of dealing with a basis in echelon form means that the sizes of the entries can increase dramatically, so in \mathbb{Q} , we cannot assume that field operations are $\mathcal{O}(1)$.

8.14.3 Applying rational reconstruction

This procedure is a good candidate for applying rational reconstruction. Finding echelon forms and applying the vector-decomposition iteration are genuinely $\mathcal{O}(n^3)$ and $\mathcal{O}(n^2)$ when working with finite fields.

Given a prime p , we reduce our basis vectors modulo p to give a basis $w_p^{(1)}, w_p^{(2)} \dots w_p^{(n)}$ of a module W_p over $\text{GF}(p)$. (If any of the denominators of the original basis vectors are divisible by p , we multiply by a scalar so that the denominators disappear.) We similarly reduce the vector $v \in W$ modulo p to give v_p . We then perform the decomposition of v_p to give an expression:

$$v_p = \sum_{i=1}^k \lambda_{i,p} w_p^{(i)} \quad (8.14.5)$$

We reconstruct λ_i from the various $\lambda_{i,p}$ for p prime. We can test whether our calculated value is correct by verifying that (8.14.2) holds. There is a delicate trade-off here, as the more primes are chosen, the longer it takes to give an estimate for λ_i but the more likely it is that the estimate is correct. Moreover, we do not want to perform the accuracy check too often, because it involves the slower arithmetic over \mathbb{Q} and because we cannot check a particular value λ_i until we have estimates for all λ_k ($1 \leq k \leq n$). We usually started the reconstruction with 10 primes, adding 5 primes each time if the accuracy check failed.

Note that reconstructing many scalars at once can be easier than reconstructing a single scalar, as we have some extra ‘hints’ as to what the denominator is likely to be. If large denominators are involved in the answer, it is usually the case that every non-integer entry has the same large denominator (possibly with some small factor removed or added). This can give us an idea as to whether our reconstruction is sensible without actually performing the check.

Example 8.15 The group $O_8^+(2)$ has a 175-dimensional representation which is a constituent of a permutation representation on 960 points. Finding the 175-dimensional submodule of $\overline{\Omega}$ is straightforward. To find the 175×175 matrices generating this representation, we tried:

- rational reconstruction with 10 primes p , $70 \leq p \leq 110$; and
- Gaussian elimination in \mathbb{Q} .

Using Gaussian elimination took 2041 seconds, whereas the reconstruction method took 142 seconds including the time to check the action. Thus reconstruction was about 14 times quicker than Gaussian elimination in \mathbb{Q} .

8.15 Finding suitable permutation representations

Usually G has several permutation representations that are useful for finding matrix representations. In this section, we will discuss how to search for suitable permutation representations.

Let H be a subgroup of G . Then G acts on the (right) cosets Hg of H by right-multiplication, which gives rise to a permutation representation of G :

$$G \rightarrow \text{Sym}(G : H) \tag{8.15.1}$$

Under this homomorphism, H gets mapped to the stabilizer of an element. Thus there is a natural correspondence between subgroups of a group and its permutation representations. Permutation representations of G are equivalent if there is a bijection between the underlying sets which preserves the action. In terms of this action, conjugate subgroups of G give rise to equivalent permutation representations, and *vice versa*.

If the full subgroup lattice of G (or the set of conjugacy classes of subgroups of G)

is available, then all the permutation representations of G can be computed by using this action of G .

Often we do not have the set of conjugacy classes of subgroups, and typically, this is expensive to compute. However, we are usually not interested in the set of all permutation representations, because most of them are far too large to deal with anyway.

To find the smaller representations, we can split the problem into two parts:

- Find the set of (small-degree) permutation characters of G .
- Find a subgroup H of G whose permutation representation gives rise to that permutation character. (In particular, we know the order of H .)

Definition 8.16 (Burnside [8]) *A table of marks for a group G is a matrix M whose rows and columns are labelled by conjugacy classes of subgroups of G , and given subgroups H, K of G in conjugacy classes H^G, K^G respectively, the (H^G, K^G) -entry of M is the number of fixed points of K in the transitive action of G on the cosets of H .*

Tables of marks have been computed for many simple groups, and are available as a data library in **GAP**. If we are fortunate enough to have a table of marks computed for G , then the first part has been effectively solved already, as it is easy to extract a complete list of permutation characters from a table of marks. Otherwise, we can use the methods of Breuer and Pfeiffer [7] to find ‘possible’ permutation characters. These are characters of G which obey a set of necessary conditions for a character to be a permutation character, and their methods have been implemented as **GAP** routines. Unfortunately, there is no known easily-computed sufficient condition.

Once we have the characters, it is usually straightforward to find a corresponding subgroup. In many cases, we know the maximal subgroups of G , and that gives a fair indication of what the structure of H has to be (or at least, where we should be looking).

8.16 Example session with Split-P

The group $U_3(3)$ has a permutation representation on 36 points given by the action on the right cosets of $L_2(7)$. The corresponding permutation module decomposes as

$$1 + 7b + 7c + 21 \quad (8.16.1)$$

We will show how Split-P can be used to decompose this module.

Firstly, we find a subgroup $L_2(7) = \langle a, b \rangle$ by random searching and find the appropriate coset action:

```
gap> G := PSU(3,3);
gap> repeat t := Random(G); until Order(t) mod 2 = 0; a := t^(Order(t)/2);
gap> repeat t := Random(G); until Order(t) mod 3 = 0; b := t^(Order(t)/3);
gap> repeat b := b^Random(G); until Index(G, Group(a,b)) = 36;
gap> G2 := Image(FactorCosetAction(G, Group(a,b)));
Group([ (1,13,19,7,4,22,10,16)(2,15,20,9,5,24,11,18)(3,14,21,8,6,23,12,17)(25,
33,30,34)(26,32,28,35)(27,31,29,36), (1,7,10,35,34,36,32,22)(2,6,3,15,27,
20,18,30)(4,24,14,25)(5,16,29,8)(9,17,26,23,11,31,12,33)(13,28,19,21) ])
gap> OrbitLengths(G2);
[ 36 ]
```

We then ask Split-P to find the intersection matrices for this representation:

```
gap> M := IntersectionMatrices(G2);
rec(
  group := Group([ (1,31,10,14,16,8,4,28)(2,32,21,34,30,18,13,17)(3,7,25,36,
29,22,12,9)(5,20,27,35)(6,11,23,33)(15,24,26,19),
(1,14,4,33,27,24,11,8)(2,36,9,32,26,21,17,20)(3,35,16,18)(5,34,12,6,29,
13,19,22)(7,31,23,10)(15,25,30,28) ]),
  matrices := [ [ [ 1, 0, 0, 0 ], [ 0, 1, 0, 0 ], [ 0, 0, 1, 0 ],
[ 0, 0, 0, 1 ] ],
[ [ 0, 1, 0, 0 ], [ 0, 0, 4, 1 ], [ 7, 0, 0, 2 ], [ 0, 6, 3, 4 ] ],
[ [ 0, 0, 1, 0 ], [ 7, 0, 0, 2 ], [ 0, 4, 0, 1 ], [ 0, 3, 6, 4 ] ],
[ [ 0, 0, 0, 1 ], [ 0, 6, 3, 4 ], [ 0, 3, 6, 4 ], [ 21, 12, 12, 12 ] ] ],
  orbitsizes := [ 1, 7, 7, 21 ], numpoints := 36,
  pairing := [ 1, 3, 2, 4 ] )
```

Note that Split-P conjugates the original group so that the suborbits consist of consecutive integers. We then ask it to find the simultaneous eigenvectors over \mathbb{Q} . One of these will give the 21-dimensional module.

```
gap> ev := FindSimultaneousEigenvectors(M.matrices);
[[ 1, 1/7, 1/7, -1/7 ], [ 1, 1, 1, 1 ] ]
```

The eigenvector we want is plainly the first one. Accordingly we ask `Split-P` for the matrices on this submodule.

```

gap> M1 := PermSubmoduleMatsRational(ev[1], M, 0, 0, [5..50]);
Finding CG-module
Prime p_1 = 11
42/42 matrix rows reconstructed, lcd = 1
rec(
  mats := [ [ [ 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
              [ 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
              [ 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
              [ 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
              [ 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
              [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
              [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
              [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
              [ 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
              [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0 ],
              [ 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
              [ 1, 0, 0, -1, 1, 1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, -1, 0, 0, 0, 0 ]
            ],
            [ 0, -1, 0, -1, 1, 0, 0, 1, 0, 0, -1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
            [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0 ],
            [ 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
            [ 0, 0, -1, -1, 0, 0, 0, 1, 1, 1, -1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
            [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0 ],
            [ -1, 0, 1, 1, -1, 0, 0, -1, -1, 0, 0, 0, -1, -1, 0, 1, -1, 1, -1, -1, 0, 0 ],
            [ 0, 0, 1, 1, 0, 0, -1, -1, -1, -1, 0, 0, -1, -1, 0, 1, -1, 0, -1, 0, 0, 0 ],
            [ [ 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
              [ 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
              [ 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
              [ 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
              [ 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
              [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
              [ 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
              [ 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
              [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0 ],
              [ 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
              [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0 ],
              [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0 ],
              [ 0, -1, 0, -1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, -1, 0, 0, 0, 0 ],
              [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 ],
              [ 0, 0, -1, -1, 0, 0, 0, 1, 1, 1, -1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
              [ 1, 0, 0, -1, 1, 0, -1, 0, 0, -1, 0, 1, 0, 0, 1, 0, 0, -1, 1, 1, -1, 0 ],
              [ 0, -1, 0, -1, 1, 0, 0, 1, 0, 0, -1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
              [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0 ]
            ]
          ]
)

```

```

],
[ 0, 1, 0, 1, -1, -1, 0, -1, -1, -1, 1, 0, -1, 0, 0, 0, -1, 0, 0,
  0, -1 ],
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0 ],
[ 0, 1, -1, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, -1
] ] ], lcd := 1 )

```

It has found two 21×21 matrices giving the irreducible representation we need. To find the 7-dimensional representations, we must look for irrational eigenvectors of the intersection matrices.

```

gap> Factors(CharacteristicPolynomial(M.matrices[3]));
[ x_1-7, x_1-1, x_1^2+4*x_1+13 ]

```

The solutions of the quadratic $x^2 + 4x + 13$ are $x = -2 \pm 3\sqrt{-1}$. Thus we extend the field to include $\sqrt{-1}$ and then find another simultaneous eigenvector:

```

gap> F := AlgebraicExtension(Rationals, x^2 + 1);;
gap> u := RootOfDefiningPolynomial(F);;
gap> N := NullspaceMat(M.matrices[3] - (-2+3*u) *IdentityMat(4));
[ [ !7, (-2+3*a), (-2-3*a), !1 ] ]

```

We then use this eigenvector to give one of the 7-dimensional representations.

```

gap> M2 := PermSubmoduleMatsQuadratic(N[1], M, F, E(4), 0);
Prime p = 31
Spinning up CG-module
skip = [ 6, 7, 8, 10, 11, 12, 13, 14 ]
Calculating matrix entries for generator 1
Calculating matrix entries for generator 2
14/14 matrix rows reconstructed, lcd = 1
rec(
  mats := [ [ [ 0, 1, 0, 0, 0, 0, 0 ], [ 0, 0, 0, 1, 0, 0, 0 ], [ 0, 0, 0, 0,
    0, 1, 0 ], [ 0, 0, 1, 0, 0, 0, 0 ], [ 0, 0, 0, 0, 0, 0, 1 ],
    [ 0, E(4), -1+E(4), 0, -1, 0, E(4) ],
    [ -1, 0, 0, -1-E(4), 0, -E(4), -E(4) ] ],
  [ [ 0, 0, 1, 0, 0, 0, 0 ], [ 0, 0, 0, 0, 0, 1, 0, 0 ],
    [ -E(4), 0, 0, 1, 0, E(4), 0 ], [ 0, -E(4), 0, E(4), 1, 0, 0 ],
    [ 0, 0, 0, 1, 0, 0, 0 ], [ 0, 1, 1, -1, E(4), 0, -E(4) ],
    [ -1, 0, 0, -1-2*E(4), -1, 1-E(4), 0 ] ] ], lcd := 1 )

```

Here $E(4)$ is the GAP notation for $\sqrt{-1}$. If we wanted the other 7-dimensional representation, we would have used $-E(4)$ instead.

If we wanted to find the direct sum of $7b$ and $7c$ (writable over \mathbb{Q}), we could have taken a vector in the 2-dimensional nullspace of $m^2 + 4m + 13$ (where m is the third intersection matrix) and used the function `PermSubmoduleMatsRational`.

Chapter 9

The 231-dimensional representations of

M_{24}

The group $G = M_{24}$ has two complex-conjugate irreducible 231-dimensional representations. In this chapter, we will construct these representations.

9.1 Permutation and tensor product depths

We define the *permutation depth* of χ to be the degree of the smallest permutation representation of G containing χ as a constituent:

$$\pi\text{-depth}(\chi) = \min\{|G : H| : H < G, \langle \chi, 1_H^G \rangle > 0\} \quad (9.1.1)$$

This quantity is finite, as $1_{\{1\}}^G$ is the regular representation which contains every irreducible character of G . The permutation depth of χ describes roughly how difficult it is to construct the representation ρ from a permutation representation of G .

Similarly, the *tensor product depth* of χ describes roughly how difficult it is to construct the representation from tensor products of other representations:

$$\otimes\text{-depth}(\chi) = \min\{\chi_i \otimes \chi_j(1) : \chi_i, \chi_j \in \text{Irr}(G) \setminus \chi^{\text{Gal}}, \langle \chi, \chi_i \otimes \chi_j \rangle > 0\} \quad (9.1.2)$$

χ	$\chi(1)$	$\pi\text{-depth}(\chi)$	$\otimes\text{-depth}(\chi)$	Constituent of
χ_2	23	24	46575	$\chi_3 \otimes \chi_{16}$
χ_3	45	40320	23805	$\chi_2 \otimes \chi_{15}$
χ_4	45	40320	23805	$\chi_2 \otimes \chi_{16}$
χ_5	231	255024	122199	$\chi_2 \otimes \chi_{23}$
χ_6	231	255024	122199	$\chi_2 \otimes \chi_{23}$
χ_7	252	276	529	$\chi_2 \otimes \chi_2$
χ_8	253	552	529	$\chi_2 \otimes \chi_2$
χ_9	483	759	5796	$\chi_2 \otimes \chi_7$
χ_{10}	770	21252	40733	$\chi_2 \otimes \chi_{18}$
χ_{11}	770	21252	40733	$\chi_2 \otimes \chi_{18}$
χ_{12}	990	159390	2025	$\chi_4 \otimes \chi_4$
χ_{13}	990	159390	2025	$\chi_3 \otimes \chi_3$
χ_{14}	1035	1288	2025	$\chi_3 \otimes \chi_3$
χ_{15}	1035	255024	1035	$\chi_2 \otimes \chi_3$
χ_{16}	1035	255024	1035	$\chi_2 \otimes \chi_4$
χ_{17}	1265	2024	5796	$\chi_2 \otimes \chi_7$
χ_{18}	1771	3542	5819	$\chi_2 \otimes \chi_8$
χ_{19}	2024	3795	2025	$\chi_3 \otimes \chi_4$
χ_{20}	2277	7590	11109	$\chi_2 \otimes \chi_9$
χ_{21}	3312	10626	11109	$\chi_2 \otimes \chi_9$
χ_{22}	3520	6072	5796	$\chi_2 \otimes \chi_7$
χ_{23}	5313	17710	5313	$\chi_2 \otimes \chi_5$
χ_{24}	5544	21252	11340	$\chi_3 \otimes \chi_7$
χ_{25}	5796	10626	11340	$\chi_3 \otimes \chi_7$
χ_{26}	10395	30360	10395	$\chi_3 \otimes \chi_5$

Table 9.1: Permutation and tensor product depths for M_{24}

where χ^{Gal} is the set of Galois conjugates of χ (we exclude these to avoid ‘self-referential’ constructions for ρ). This quantity is finite, because if χ_0 is a faithful character, then every irreducible character of G is a constituent of χ_0^n for some $n \in \mathbb{N}$.

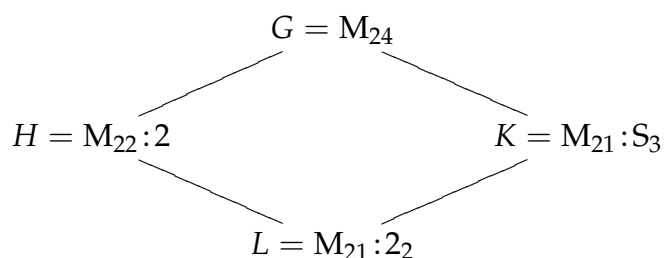
Let ρ be one of the 231-dimensional irreducible representations of M_{24} and let χ be its character. We have that:

$$\pi\text{-depth}(\chi) = 255024, \quad \otimes\text{-depth}(\chi) = 122199$$

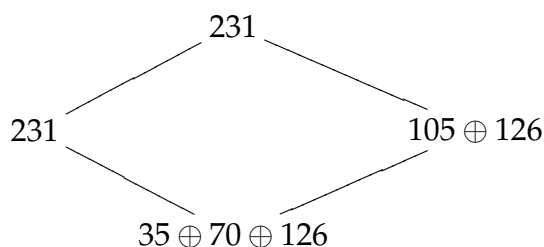
which are both rather high. (In fact, Table 9.1 shows that in some sense ρ is the hardest representation of M_{24} to construct by the usual methods.)

9.2 The amalgam

Instead of trying to decompose a tensor product or permutation representation, we decided to construct ρ by amalgamation. We used the following amalgam:



In the natural representation on 24 points, the group $M_{22}:2$ is the stabilizer of a 2-set, $M_{21}:S_3$ is the stabilizer of a 3-set, and $M_{21}:2_2$ is the stabilizer of a 3-set and one of the points therein. The character tables in **GAP** were used to see how the representation ρ restricts to the subgroups H , K and L :



9.3 The restrictions of ρ

9.3.1 Constructing $\rho|_H$

The 231-dimensional representation of $H = M_{22} : 2$ is relatively easy to construct, as it is a constituent of a permutation representation of H on 924 points. We must make sure that we take the $+$ -type representation rather than the $-$ -type one, although it is easy to convert between the two if we find the wrong one (both standard generators of $M_{22} : 2$ are outer elements, so we need to take the negatives of both the generators).

9.3.2 Constructing $\rho|_K$

The 105-dimensional representation of $K = M_{21} : S_3$ can be easily constructed from one of the permutation representations on 168 points. Again, we must make sure that we take the $+$ -type rather than the $-$ -type.

The 126-dimensional representation of K is harder. It is a constituent of a permutation representation on 960 points, but since we need to work in the quadratic extension $\mathbb{Q}(\sqrt{-15})$, decomposing this representation is difficult and leads to very complicated matrices.

We obtained better matrices by considering the subgroup $K_1 = 2^4 : (3 \times A_5) : 2$ of index 21 in K . This has a 6-dimensional representation (actually a representation of $\overline{K_1} = (3 \times A_5) : 2$) which we can induce to the required 126-dimensional representation of K . Finding the 6-dimensional representation of K_1 is straightforward; we found it from a permutation representation of $\overline{K_1}$.

Note that there are actually two complex-conjugate 126-dimensional representations. The choice of which one to use determines which 231-dimensional representation of M_{24} we obtain.

9.4 Finding the intersection L

Words for standard generators of the subgroup $L_H \cong L = M_{21} : 2$ in terms of standard generators c, d of H can be found in the Web Atlas [57]:

$$g_1 = c \tag{9.4.1}$$

$$h_1 = ((cddcd)^{-2})^{ddcdcd} \tag{9.4.2}$$

(We have taken the inverse of the second generator for consistency with our calculations in G later.)

Words for standard generators of the subgroup $L_K \cong L$ in terms of standard generators e, f of K were found by the methods described in my MPhil thesis [34]:

$$g_2 = e \tag{9.4.3}$$

$$h_2 = ((efefeff)^3)^{eff} \tag{9.4.4}$$

9.5 Standard basis for standard generators of L

Let g, h denote standard generators of the 231-dimensional representation of L under consideration (decomposing as $35 \oplus 70 \oplus 126$). Define:

$$M_1 = g + h + gh^3 - 1 \tag{9.5.1}$$

$$M_2 = g + h + gh^2 \tag{9.5.2}$$

$$M_3 = (gh)^{(ghh)^4}gh + ghghhgh + (ghghhgh)^{-1} + 1 \tag{9.5.3}$$

These matrices have nullity 1. Let v_i ($1 \leq i \leq 3$) denote arbitrary non-zero vectors in their respective nullspaces, and let V_i denote the vector subspace of $V \cong \mathbb{C}^{231}$ obtained

by spinning up v_i . Then:

$$V = V_1 \oplus V_2 \oplus V_3 \quad (9.5.4)$$

and the spaces V_i have dimensions 35, 70 and 126 respectively. This defines a standard basis of V . With respect to this basis, elements of L are in block-diagonal form.

Let S_1 be the change of basis matrix with respect to g_1 and h_1 , and define:

$$c' = S_1 c S_1^{-1}, \quad d' = S_1 d S_1^{-1} \quad (9.5.5)$$

Similarly, let S_2 be the change of basis matrix with respect to g_2 and h_2 , and define:

$$e' = S_2 e S_2^{-1}, \quad f' = S_2 f S_2^{-1} \quad (9.5.6)$$

9.6 Calculations in G

We define the following groups:

$$G_0 = \langle a_0, b_0 \rangle \cong M_{24}$$

$$H_0 = \langle c_0, d_0 \rangle \cong M_{22} : 2$$

$$K_0 = \langle e_0, f_0 \rangle \cong M_{21} : S_3$$

$$L_0 = K_0 \cap H_0 = \langle g_0, h_0 \rangle \cong M_{21} : 2$$

where:

$$a_0 = (1, 7)(2, 17)(3, 4)(5, 13)(6, 9)(8, 15)(10, 19)(11, 18)(12, 21)(14, 16)(20, 24)(22, 23)$$

$$b_0 = (1, 21, 14)(2, 9, 15)(3, 4, 6)(5, 18, 10)(13, 17, 16)(19, 24, 23)$$

$$c_0 = (1, 2)(5, 24)(6, 16)(7, 9)(12, 13)(14, 18)(15, 23)(21, 22)$$

$$d_0 = (1, 2)(3, 20, 23, 16)(4, 18)(5, 19, 22, 8)(6, 9, 12, 24)(7, 21)(10, 17, 13, 14)(11, 15)$$

$$e_0 = c_0$$

$$f_0 = (1, 3, 2)(4, 5, 21)(6, 22, 17)(7, 18, 8)(9, 16, 13)(10, 19, 24)(11, 15, 20)(12, 14, 23)$$

$$g_0 = c_0$$

$$h_0 = (4, 12, 14, 19, 11)(5, 6, 8, 24, 23)(9, 15, 17, 18, 22)(10, 20, 21, 16, 13)$$

These permutations were chosen so that g_0 and h_0 satisfy the equations given for g_i and h_i in section 9.4 above. This means that we must have the correct amalgam.

9.7 Completing the amalgam

Let x, y, z be indeterminates, and let:

$$X = \text{diag}(\underbrace{x, x, \dots, x}_{35 \text{ times}}, \underbrace{y, y, \dots, y}_{70 \text{ times}}, \underbrace{z, z, \dots, z}_{126 \text{ times}}) \quad (9.7.1)$$

Any element commuting with the subgroup L is of this form, so let $e'' = Xe'X^{-1}$, $f'' = Xf'X^{-1}$. We wish to find x, y and z such that:

$$\langle c', d', e'', f'' \rangle \cong M_{24} \quad (9.7.2)$$

We may pick any value for z , because this part commutes with e and f too. Without loss of generality, we pick $z = 1$. Similarly, conjugating by a scalar has no effect, so we

may assume without loss of generality that $y = 1$. So we only have one unknown left to determine.

By observing cycle shapes, we see that $d_0 f_0$ is an $8A$ -element, so that in the 231-dimensional representation df should have trace -1 . This yields the equation:

$$\text{Tr}(df') = \left(-\frac{3}{4} - \frac{1}{8}\eta\right) + \frac{15}{368}x^{-1} = -1 \quad (9.7.3)$$

where $\eta = (1 + \sqrt{-15})/2 = \mathfrak{b}15$. The solution is:

$$x = -\frac{3}{92}(\eta + 3) \quad (9.7.4)$$

9.8 Finding standard generators

We currently have four 231×231 matrices c' , d' , e'' and f'' which together generate M_{24} . We wish to find standard generators a and b . Working in this representation is rather unwieldy, so we switch our calculations to the permutations on 24 points defined above. The words we found are:

$$a = (c'd'^2c'd')^5; \quad (9.8.1)$$

$$b = ((c'd'^2)^2)^{e''f''e''f''e''f''e''f''}; \quad (9.8.2)$$

The matrix $M = a + b + b(ab^2)^2 - 1$ has nullity 1 and can be used to reseed this representation.

Chapter 10

Miscellaneous techniques

While most of the representations in the shortened Hiss-Malle list can be found using our Split-P system or by using the generic constructions given in Chapter 7, there are some representations where other techniques were useful. In this chapter, we describe some of these techniques.

10.1 Elementary techniques

10.1.1 Restriction

If $\rho : G \rightarrow \mathrm{GL}_n(\mathbb{C})$ is a representation of G , then for $H \leq G$, $\rho|_H$ is a representation of H . In some cases, we can find a useful irreducible representation of H in this manner.

Example 10.1 *Let $H = {}^2F_4(2)'$, the Tits group. This group is contained in the group $G = 2.F_4(2)$, and G has a 52-dimensional representation described in the ATLAS. By restricting to H , we get the reducible representation 26ab, which can be decomposed to get two irreducibles 26a and 26b.*

Example 10.2 *The 45-dimensional and 231-dimensional representations of M_{23} can be found by restricting representations of M_{24} .*

Example 10.3 *We have seen in section 7.1 that the representations of A_n can be found by restricting representations of S_n (and splitting, in the case of a representation indexed by a self-dual partition).*

If no splitting is involved, the computational work involved here is in finding standard generators of H as words in the standard generators of G . This is straightforward providing G has a ‘nice’ representation in which to work.

10.1.2 Induction

If $\sigma : H \rightarrow GL_n(\mathbb{C})$ is a representation and H is a subgroup of G with index r , then σ can be *induced* to a representation $\rho : G \rightarrow GL_{rn}(\mathbb{C})$. If n is fairly small (say 1 or 2), then the induced representation is sometimes useful for our purposes.

Example 10.4 *The group $G = L_3(5)$ has a subgroup $H = 5^2 : GL_2(5)$ with index 31. The subgroup H has 10 equivalence classes of 4-dimensional representations (note that the normal subgroup 5^2 is always in the kernel of these representations), and these can be induced to give the 10 equivalence classes of 124-dimensional representations of $L_3(5)$.*

10.1.3 Algebraic conjugates

Some representations have several algebraic conjugates. It is straightforward in GAP to apply a field automorphism to each matrix entry in a representation to get an algebraically conjugate representation. The easiest example is the case of a non-real representation, which has a complex conjugate representation which is not equivalent.

10.1.4 Automorphisms

Applying a group automorphism to a representation can sometimes give a different representation. For instance, there are 3 representations with genus $(O_8^+(2), 35, +)$ and they are permuted by the outer automorphism group S_3 of $G = O_8^+(2)$. If find an

explicit automorphisms of order 3 in $\text{Out}(G)$, then we can use it to write down the other 2 representations given any one.

10.2 Decomposing tensor products

Given two representations ρ, σ of a group G , we can construct the *tensor product representation* $\rho \otimes \sigma$. The construction is straightforward: take a set of generators for G , and for each generator x form the Kronecker product $\rho(x) \otimes \sigma(x)$. The character of this representation is simply the product of the characters of ρ and σ . We can therefore use the character table of G to write $\rho \otimes \sigma$ as a direct sum of irreducible representations. Usually, the tensor product representation is not irreducible. However, we can sometimes use some of the techniques in this section to decompose the representation into irreducibles.

We can also exploit the fact that the tensor square $\rho \otimes \rho$ can always be decomposed into symmetric and anti-symmetric parts. Matrices corresponding to these parts are also straightforward to find, and this can save some effort. Moreover, if G has a proper cover \bar{G} , then we can use representations of \bar{G} in the construction as well. This is frequently useful if \bar{G} has representations of smaller dimension than G .

10.2.1 Plesken-Souvignier splitting

Plesken and Souvignier [40] describe a way of decomposing a rational representation into homogeneous parts. We have written a program in C [28] which uses their technique.

Let V be a QG-module which decomposes into irreducibles:

$$V = m_1 V_1 \oplus m_2 V_2 \oplus \cdots \oplus m_k V_k \quad (10.2.1)$$

Let χ_i denote the character afforded by the irreducible module V_i . We have a QG-

homomorphism $\rho : \mathbb{Q}G \rightarrow \text{End}_{\mathbb{Q}}(V)$: we will denote images under this map with a bar.

Consider the map:

$$\begin{aligned} \theta : \overline{\mathbb{Q}G} &\rightarrow \overline{\mathbb{Q}G} \\ u &\mapsto \frac{1}{|G|} \sum_{g \in G} u^g \end{aligned} \tag{10.2.2}$$

Then for $h \in G$, $\theta(\bar{h})$ is an element of $Z(\text{End}_{\mathbb{Q}G}(V))$. It has eigenvalues $\chi_i(h)/\chi_i(1)$ with multiplicity $m_i\chi_i(1)$, and the corresponding eigenspaces are the direct summands $m_i V_i$ in (10.2.1).

The main idea of [40] is that the map θ can be found as the limit of some approximations which are easier to calculate. Let S be a generating set for G and define:

$$\begin{aligned} \theta_S : \overline{\mathbb{Q}G} &\rightarrow \overline{\mathbb{Q}G} \\ u &\mapsto \frac{1}{|S|} \sum_{g \in S} u^g \end{aligned} \tag{10.2.3}$$

Then $(\theta_S)^n$ converges to θ inside $\text{End}_{\mathbb{C}}(V)$.

To calculate $\theta(\bar{x})$, we perform the iteration:

$$v_n = (\theta_S)^n(\bar{x}) = \theta_S(v_{n-1}) \tag{10.2.4}$$

At various points in the iteration, we use a continued fraction expansion of the entries of v_n to guess what the limit is. We can then test our guess by checking whether the matrix commutes with the group action. The iteration is not done using exact arithmetic because of the problem of intermediate expression swell (which drastically affects the speed of the iteration). Instead we use floating point arithmetic. This is likely to cause problems if the common denominator for $\theta(\bar{x})$ is very large.

Example 10.5 *During this iteration in a particular case, one of the matrix entries is 0.133329856, which has a continued fraction expansion:*

$$\frac{1}{7 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1277 + \dots}}}} \quad (10.2.5)$$

We guess that 1277 is an error term and truncate the expansion at the previous term. This gives an estimate of $2/15$. This turns out to be the correct answer.

There are other matrices which can perform the function of θ_5 , some of which converge to θ faster.

Program Rho

We have written a program **Rho** which implements the Plesken-Souvignier technique to split a representation. Early in our investigation, we used this to construct the representations of $U_4(2)$ (reproducing the results in the Plesken-Souvignier paper [40]), but these representations were later replaced by those obtained from permutation representations (as these tended to have better sparsity).

Our program extends the basic method to work with fields $\mathbb{Q}(\alpha)$ where $\alpha \in \mathbb{C} \setminus \mathbb{R}$ and α is quadratic over \mathbb{Q} . The method is the same except at the limit guessing stage. Because $\{1, \alpha\}$ forms a \mathbb{R} -basis for \mathbb{C} , we can write any entry of the matrix v_n as $r + \alpha s$, and we can perform continued fraction recognition for r and s separately. We were not able to find a way to cope with real extensions of \mathbb{Q} .

When **Rho** was written, **GAP** had no support for floating point arithmetic.¹ We

¹The latest release (4.4.6) of **GAP** has some support for this, but we have not had the opportunity to update our code to use this functionality.

therefore performed almost all the calculations in C, providing a simple wrapper for GAP.

Problems

One significant problem with this method of finding irreducible representations is that the operation of tensoring means the matrices get large very quickly unless G has a number of small representations. For many groups, the tensor product matrices are too large for practical computations.

10.2.2 MeatAxe methods

The methods of Parker's MeatAxe [36] can be adapted to work with representations in extensions of \mathbb{Q} [37, 23].

Spinning up

Let V be a $\mathbb{C}G$ -module with an irreducible submodule W . Suppose $w \in W$. Then we can find a basis for W by *spinning up* the vector w by the action of G .

The algorithm is as follows. Let $S = \{g_1, \dots, g_n\}$ be a set of generators for S . We maintain a list L of linearly independent vectors known to be in W , and we stop when L spans W . Initially, L contains just the vector w . Set $k = 1$ to start.

Look at the k th element of L and construct the vectors wg_i for $1 \leq i \leq n$. If wg_i is linearly independent of all the vectors in L , then add it to the list. Otherwise, carry on. After considering all values of i , increment the value of k .

The algorithm stops either when L contains $\dim W$ vectors, or when $k > |L|$ (because then L is closed under the action of the generating set S , and so it must span a $\mathbb{C}G$ -submodule).

10.2.3 Decomposing large modules

Suppose we are interested in finding a particular rational representation ρ of the group G . Calculations with the character table of G show that the representation can be found by decomposing a larger rational representation σ which we are already able to construct (say, a large tensor product representation). Splitting the representation over the rationals is extremely difficult. In particular, our methods for dealing with general matrix representations are not as powerful as our methods for decomposing permutation representations, so we are restricted in terms of the maximum dimension of σ that we can deal with.

However, we can try the following idea. Take a set of primes P , and for each $p \in P$, calculate the representation σ_p , which is σ reduced modulo p . We can then use the MeatAxe [36] to find the representation ρ_p , which is equivalent to ρ modulo p . We then find a common seed vector for all the ρ_p and reseed them all to new representations $\tilde{\rho}_p$. These representations are compatible, and they are equal to the reductions of a representation $\tilde{\rho}$ modulo p . We can then try to reconstruct $\tilde{\rho}$ (which is equivalent to ρ , the representation we wanted to find) by reconstructing each matrix entry from the matrix entries of the $\tilde{\rho}_p$. Note that it is harder to test whether the reconstructed representation $\tilde{\rho}$ is correct; we would need a presentation for G to be absolutely sure that we had not introduced any mistakes.

10.2.4 Floating point rational reconstruction

Another possibility that was briefly considered was that of ‘floating point’ (as opposed to p -adic) rational reconstruction. Floating point arithmetic is not exact, but it is $\mathcal{O}(1)$. We could perform our rational calculations using the computer’s floating point arithmetic and then use continued fraction expansions to find a rational which is close to the floating point number calculated. This approach was rejected as being too difficult

to analyse and unlikely to scale well. There is in general no way of predicting how large the denominators were likely to be in the final answer, and the larger the denominators, the more bits the floating point representation needed to have. With the p -adic methods, we could at least add more primes if the denominators were proving to be too large, but with floating point arithmetic, this involves a lot more work and some careful numerical analysis which we did not feel qualified to attempt.

10.3 Dixon's method

John Dixon invented a method for producing representations affording a character χ of a group G subject to the condition that there exists a subgroup $H \leq G$ such that χ_H has a constituent with degree 1 and multiplicity 1 [16]. Unfortunately, finding such a subgroup H is not easy in general (it seems to require computing the whole lattice of subgroups), and there exist groups where no such subgroup exists.

Dixon's method has been investigated by Dabbaghian-Abdoly [13] and implemented by him as a GAP package `Repsn`. The method is quite slow and memory-intensive, but it succeeded where other methods failed (notably for some small representations of the unitary groups).

Chapter 11

A database of group representations

We present our results as a database of group representations. This database is available on the CD-ROM attached to this thesis (see Appendix C). We intend to include these representations in the Web Atlas [57] in the near future.

11.1 Information supplied

For each group $G \in \mathcal{L}_0^G$, we present some or all of the following information:

- Definitions of standard generators for G and its covers (either taken from the Web Atlas or from Part I of this thesis).
- Checkers and finders for G (see Part I).
- Complex matrix representations of G with dimension at most 250 (given in terms of the matrices of standard generators of G).

For each matrix representation ρ supplied, we give:

- The name of the group G generated.
- The dimension of ρ .
- The Frobenius-Schur indicator of ρ .

- The character of ρ , usually identified with a particular row in the ATLAS or in the GAP character table library. Sometimes there is some ambiguity; this is mentioned in the notes.
- A ring over which the matrix entries can be written.
- Some data about the sparsity of the matrices and the sizes of the matrix entries.
- A note about how the representation was constructed, or the source if we have used another researcher's construction.

The database is maintained by means of a number of Perl [55] scripts which have evolved over a period of roughly two years.

11.2 Checking the data

Large databases almost inevitably include errors. We wished to minimise the number by performing a battery of tests on our data. We designed the tests so that they could be run overnight without human intervention, presenting a report on which representations passed and which failed. Those that failed were corrected until all tests passed.

11.2.1 Checker test (semi-presentation)

All our representations were on standard generators. We were therefore able to make use of the semi-presentations we calculated in Part I. We applied the checkers derived from our semi-presentations to the representations we calculated to check whether the generators we had were correct.

In about half a dozen cases, our representations failed the check, showing that we had taken the wrong generators. These cases were corrected.

11.2.2 CCL test

One common type of error in our data was mislabelling of the representations; in particular, associating them with an incorrect character. This usually occurred when there was more than one irreducible character of the same dimension.

To reduce the risk of this problem, we checked our generators with some BBOX programs. These computed some conjugacy class representatives and checked their traces. If the traces agreed with the character values, then the representation passed the test. The BBOX programs were written with the help of the conjugacy class representatives contained in the Web Atlas. Note that we do not usually check all the conjugacy classes of the representation, and we only perform this test when GAP has a character table for the group.

In some cases, there was no word program to produce conjugacy class representatives. In this case, we specified our conventions as to which conjugacy class was which in the notes. Our conventions are generally not as strict as some of the Web Atlas conventions (so, for example, we did not demand compatibility with the Atlas of Brauer Characters [27]).

Example 11.1 *For the group $S_4(5)$, we need to be able to distinguish $13a/b$, $65a/b$, $78a/b$, $104a/b/c$ and $208a/b$. One way of distinguishing these representations is by looking at their traces on class 30A. This class and class 30B are the only classes of elements of order 30, and they fuse under an outer automorphism of $S_4(5)$. Thus without loss of generality we say $(xy)^5xy^2xy^2xyxy^2$ is in class 30A, allowing us to distinguish all low-dimensional irreducible representations of $S_4(5)$.*

11.2.3 Speeding up the tests

Certain portions of the tests are extremely slow to run with some of our representations. In particular, it is quite slow to check the order of a matrix with entries which

are large integers or algebraic numbers. One way to speed the test is to use the ‘vector order’, which is an approximation to the true order of a matrix.

Given $G \leq \text{GL}_d(\mathbb{C})$, we choose a vector v of length n with arbitrary non-zero entries (we typically chose integers in $\{-10, \dots, -4, 4, \dots, 10\}$) and perform the following iteration:

$$v_0 = v; \quad v_{i+1} = v_i g \tag{11.2.1}$$

stopping when $v_m = v$ for some $m > 0$. This value m is called the *vector order* of g (with respect to v). The vector order m divides, and is usually equal to, the true order $o(g)$.¹ Instead of checking the order of g (which is expensive), we check the vector order m of g .

Another way of speeding up the tests which we did not use was to reduce the generators modulo p before starting. We did not wish to use this method with semi-presentations as so many of the representations were constructed using rational reconstruction, and we wished to test whether this process had been successful. However, it would have made certain of the conjugacy class tests significantly faster.

11.3 Scope for further work

The accompanying CD-ROM contains over 650 representations, 120 checkers and 50 finders, catalogued and presented for inclusion in the Web Atlas. It is hoped that some of these representations will prove useful to those studying group computations.

We conclude this thesis with some remarks about possible directions for extending this work.

¹We have only come across one case where it failed to hold, and this was when n was very small and when the entries of v were also small.

11.3.1 Missing representations

We were not able to construct every representation with genus in \mathcal{L}_0 . Appendix A gives a list of all representations with genera in \mathcal{L}_0 . Those which we were unable to construct are given in the right-hand column. Those which we were only able to find as a constituent of some other reducible representation are marked in bold.

A worthy project would be to construct the remaining representations. These representations cannot be readily found from permutation representations or decomposing tensor products. Some of them might succumb to the method of amalgamation from Chapter 9.

11.3.2 Dealing with irrational representations

Most of our work has dealt with rational (indeed, integral) representations of the simple groups. However, some of the more interesting and useful representations require entries from extensions of \mathbb{Q} . While we were often able to deal with quadratic extensions of \mathbb{Q} , we never tried to look at higher order field extensions. This is partly due to software difficulties (for example, while **GAP** knows that $\sqrt{-1} = \exp(\pi i/2)$, it is currently not able to evaluate $\sqrt{\exp(\pi i/2)}$). It is possible that higher order field extensions can be dealt with by using quotients of polynomial rings. For example, in order to accommodate the irrational number $\sqrt[3]{2}$ in **GAP**, we can use the field isomorphism:

$$\mathbb{Q}(\sqrt[3]{2}) \cong \mathbb{Q}[t]/(t^3 - 2) \quad (11.3.1)$$

This mechanism could also deal with irrational elements which cannot be expressed in terms of radicals, and this would certainly be necessary when dealing with field extensions of degree 5 and higher. Such a system would require major surgery to the programs we used, and it remains to be seen whether the resulting representations

would be useful.

11.3.3 Better bases

Two representations $\rho, \sigma : G \rightarrow \mathrm{GL}_d(\mathbb{C})$ are equivalent if and only if there exists $g \in \mathrm{GL}_d(\mathbb{C})$ such that $\rho(h) = g\sigma(h)g^{-1}$ for all $h \in G$. However, not all representations in the same equivalence class are equally good for computation. Ideally, we would like our representing matrices to be sparse, writeable over a small field (preferably a small ring of algebraic integers or even \mathbb{Z}) and with small entries. Some of the representations we provide are not optimal with respect to these criteria, and it would be good to have these representations written with respect to a better basis.

11.3.4 Testing the rest of the Web Atlas

In section 2.3, we tested the representations of the sporadic groups and their automorphism groups in the Web Atlas with the semi-presentations found in that chapter. It would be worthwhile to use the other semi-presentations computed in Part I to test the other representations in the Web Atlas.

11.3.5 Larger class of groups

It would be good to be able to extend this work to a larger class of groups than the simple groups. The Hiss-Malle paper [21] classifies the low dimensional representations of quasisimple groups (that is, perfect groups G such that $G/Z(G)$ is simple), and it would be useful to have these representations constructed. We have made a start on this project, and some of these representations are available on the accompanying CD-ROM. For many of these representations, the techniques we have used in this thesis will not be sufficient. In particular, permutation representations on small numbers of points (which proved to be so useful in the simple group case) are often not faithful in the quasi-simple group case.

We could also consider the class of almost simple groups. Many of these representations can be found using the techniques of this thesis (and in particular, small degree permutation representations are often useful).

The Web Atlas covers bicyclic extensions of simple groups. Representations of such groups will generally be at least as hard to construct as those of quasisimple groups. To construct all these representations would be quite an ambitious project.

11.3.6 Hard cases

The group $L_3(4)$ probably deserves a research project of its own! Its Schur multiplier is $3 \times 4 \times 4$ and its outer automorphism group is D_{12} . It thus has a very large number of bicyclic extensions, and a corresponding large number of representations to find. Similarly, the group $U_4(3)$ has Schur multiplier $3 \times 3 \times 4$ and outer automorphism group D_8 .

11.3.7 Extending coverage for the sporadic groups

The sporadic groups are perhaps the most interesting of the finite simple groups, and so it makes sense to extend the dimension limit of 250 to a larger one. An interesting extension of this project would be:

- Find all complex irreducible matrix representations of the sporadic groups and their bicyclic extensions up to dimension 1000.
- Find the smallest (faithful) complex irreducible matrix representation of each sporadic groups (possibly excepting the Monster group \mathbb{M}).

Some of these representations are probably inaccessible without developing new techniques for dealing with them.

Appendices

Appendix A

Table of representations

Here we give a complete list of representations whose representation type is in \mathcal{L}_0 , together with information about whether we have constructed them.

- Representations which we have constructed and which are available on the CD-ROM are listed in the ‘Available’ column.
- Reducible representations which we have constructed and whose irreducible constituents are *not* available separately are listed in **bold**.
- Irreducible representations which are not available and which are not constituents of any other available representation are listed in the ‘Missing’ column.
- If we do not know how many representations are of a certain representation type, we mark the representation with a question mark (*e.g.* $157a \cdots ?$ indicates that all the representations of dimension 157 are missing).

Table A.1: Representations available

Group	Available	Missing
M ₁₁	10 <i>a</i> , 10 <i>b</i> , 10 <i>c</i> , 10 <i>bc</i> , 16ab , 44, 45, 55	153 <i>a</i> , 153 <i>b</i>
M ₁₂	11 <i>a</i> , 11 <i>b</i> , 16 <i>a</i> , 16 <i>b</i> , 16 <i>ab</i> , 45, 54, 55 <i>a</i> , 55 <i>b</i> , 55 <i>c</i> , 66, 99, 120, 144, 176	
M ₂₂	21, 45 <i>a</i> , 45 <i>b</i> , 55, 99, 154, 210, 231	
M ₂₃	22, 45 <i>a</i> , 45 <i>b</i> , 230, 231 <i>a</i> , 231 <i>b</i> , 231 <i>c</i>	
M ₂₄	23, 45 <i>a</i> , 45 <i>b</i> , 231 <i>a</i> , 231 <i>b</i>	
HS	22, 77, 154 <i>a</i> , 154 <i>b</i> , 154 <i>c</i> , 175, 231	
McL	22, 231	
Co ₃	23	
Co ₂	23	
J ₂	14 <i>a</i> , 14 <i>b</i> , 14 <i>ab</i> , 21 <i>a</i> , 21 <i>b</i> , 21 <i>ab</i> , 36, 63, 70ab , 90, 126, 160, 175, 189ab , 224ab , 225	
Suz	143	
Fi ₂₂	78	
He	51ab	
HN	133 <i>a</i> , 133 <i>b</i>	
Th	248	
J ₁	56ab , 76 <i>a</i> , 76 <i>b</i> , 77 <i>a</i> , 77bc , 120abc , 133 <i>a</i> , 133bc , 209	
J ₃	85ab	
A ₅	3 <i>a</i> , 3 <i>b</i> , 5	
A ₆	8 <i>a</i> , 8 <i>b</i> , 9, 10	
A ₇	10 <i>a</i> , 10 <i>b</i> , 10 <i>ab</i> , 14 <i>a</i> , 14 <i>b</i> , 15, 21, 35	
A ₈	14, 20, 21 <i>a</i> , 21 <i>b</i> , 21 <i>c</i> , 21 <i>bc</i> , 28, 35, 45 <i>a</i> , 45 <i>b</i> , 45 <i>ab</i> , 56, 64, 70	
A ₉	21 <i>a</i> , 21 <i>b</i> , 27, 28, 35 <i>a</i> , 35 <i>b</i> , 42, 48, 56, 84, 105, 120, 162, 168, 189, 216	
A ₁₀	35, 36, 42, 75, 84, 90, 126, 160, 210, 224 <i>a</i> , 224 <i>b</i> , 225	
A ₁₁	44, 45, 110, 120, 126 <i>a</i> , 126 <i>b</i> , 126 <i>ab</i> , 132, 165, 210, 231	
A ₁₂	54, 55, 132, 154, 165	
A ₁₃	65, 66, 208, 220	
A ₁₄	77, 78	
A ₁₅	90, 91	
A ₁₆	104, 105	
A ₁₇	119, 120	

Continued on next page...

Table A.1 — Continued

Group	Available	Missing
A ₁₈	135, 136	96a . . . j
A ₁₉	152, 153	
A ₂₀	170, 171	
A ₂₁	189, 190	
A ₂₂	209, 210	
A ₂₃	230, 231	
L ₃ (3)	12, 13, 16abcd , 26a, 26b, 26c, 26bc, 27, 39	
L ₃ (4)	20, 35a, 35b, 35c, 45a, 45b, 63a, 63b, 64	
L ₃ (5)	30, 31a, 31b, 31c, 124a, 124b, 124c, 124d, 124e, 124f, 124g, 124h, 124i, 124j, 125, 155a, 155b, 155c, 186	
L ₃ (7)	56, 57, 152a, 152b, 152c	
L ₃ (8)	72, 73a, 73b, 73c, 73d, 73e, 73f	
L ₃ (9)	90, 91a, 91b, 91c, 91d, 91e, 91f, 91g	
L ₃ (11)	132, 133a, 133b, 133c, 133d, 133e, 133bcde, 133f, 133g, 133h, 133i, 133fghi	
L ₃ (13)	182, 183a, 183b, 183c, 183bc	
L ₄ (3)	26a, 26b, 39, 52, 65a, 65b, 90, 234a, 234b	
L ₄ (4)	8, 85a, 85b, 189a, 189b, 189ab	
L ₄ (5)	155	
L ₅ (2)	30, 124, 155, 217	
L ₅ (3)	120, 121	
L ₆ (2)	62, 217	
L ₇ (2)	126	
S ₄ (4)	18, 34a, 34b, 50a, 51a, 51b, 51ab, 51c, 51d, 51cd, 85a, 85b, 153, 204a, 204b, 204ab, 204c, 204d, 204cd, 225abcd	150a, 150b
S ₄ (5)	13a, 13b, 40, 65a, 65b, 78a, 78b, 90, 104a, 104b, 104c, 130, 156, 208ab	
S ₄ (7)	25a, 25b, 126, 175a, 175b, 224	
S ₄ (8)	196	
S ₄ (9)	41a, 41b	
S ₄ (11)	61a, 61b, 61ab	
S ₄ (13)	85ab	

Continued on next page...

Table A.1 — Continued

Group	Available	Missing
S ₄ (17)	145ab	171a, 171b 238
S ₄ (19)	181ab	
S ₆ (2)	7, 15, 21a, 21b, 27, 35a, 35b, 56, 70, 84, 105a, 105b, 105c, 120, 168, 189a, 189b, 189c, 210a, 210b, 216	
S ₆ (3)	13a, 13b, 78, 91a, 91b, 105, 168, 195	
S ₆ (5)	63a, 63b	
S ₆ (7)		
S ₈ (2)	35, 51, 85, 119, 135	
S ₈ (3)	41a, 41b	
S ₁₀ (2)	155, 187	
S ₁₀ (3)	121a, 121b	
U ₃ (3)	6, 7a, 7b, 7c, 7bc, 14, 21a, 21b, 21c, 21bc, 27, 28a, 28b, 28ab, 32a, 32b, 32ab	
U ₃ (4)	12, 13a, 13b, 13c, 13d, 39a, 39b, 52a, 52b, 52c, 52d, 64, 65a, 65bcde , 75abcd	
U ₃ (5)	20, 21, 28a, 28b, 28c, 84, 105, 125, 126a, 126bc , 144ab	
U ₃ (7)	43a, 43b, 43c	42, 43d . . . g
U ₃ (8)	57a, 57b, 57ab, 133a, 133b, 133c	56
U ₃ (9)	73a	72, 73b . . . i
U ₃ (11)	111a	110, 111b, 111c
U ₃ (13)		156, 157a . . . ?
U ₃ (16)		240, 241a . . . ?
U ₄ (2)	5a, 5b, 6, 10a, 10b, 15a, 15b, 20, 24, 30a, 30b, 30c, 40a, 49b, 40ab, 45a, 45b, 60, 64, 81	51a . . . d 104a, 104b
U ₄ (3)	21, 35a, 35b, 90, 140, 189, 210	
U ₄ (4)	52, 221ab	
U ₄ (5)	105	
U ₅ (2)	10, 11a, 11b, 44, 55a, 55b, 55c, 55d, 66a, 66b, 110a, 110b, 110c, 110d, 110e, 110de, 120, 165, 176, 220a, 220b, 220c, 220d, 220ab, 220cd	
U ₅ (3)		60, 61a . . . ?
U ₅ (4)		204a . . . ?
U ₆ (2)	22, 231	

Continued on next page...

Table A.1 — Continued

Group	Available	Missing
U ₆ (3)		182, 183 <i>a</i> . . . ?
U ₇ (2)		42, 43 <i>a</i> . . . ?
U ₈ (2)		85 <i>a</i> . . . ?, 86 <i>a</i> . . . ?
U ₉ (2)		170
O ₇ (3)	78, 91, 105, 168, 182, 195	
O ₈ ⁻ (2)	34, 51, 84, 204 <i>a</i> , 204 <i>b</i>	
O ₈ ⁺ (2)	28, 35 <i>a</i> , 35 <i>b</i> , 35 <i>c</i> , 50, 84 <i>a</i> , 84 <i>b</i> , 84 <i>c</i> , 175, 210 <i>a</i> , 210 <i>b</i> , 210 <i>c</i>	
O ₈ ⁻ (3)	246	
O ₁₀ ⁻ (2)	154, 187	
O ₁₀ ⁺ (2)	155, 186	
G ₂ (3)	14, 64ab , 78, 91 <i>a</i> , 91 <i>b</i> , 91 <i>c</i> , 104, 168, 182 <i>a</i> , 182 <i>b</i>	
G ₂ (4)	65, 78	
G ₂ (5)		124
Sz(8)	14 <i>a</i> , 14 <i>b</i> , 35abc , 64, 65 <i>a</i> , 65 <i>b</i> , 65 <i>c</i>	
Sz(32)	124 <i>a</i> , 124 <i>b</i>	
³ D ₄ (2)	26, 52, 196	
³ D ₄ (3)		219
² F ₄ (2)'	26 <i>a</i> , 26 <i>b</i> , 26 <i>ab</i> , 27 <i>a</i> , 27 <i>b</i> , 78	

Appendix B

BBOX: a language for black box algorithms

The BBOX language is a simple language for implementing black box algorithms for groups. Its syntax is designed to be as close as possible to that of the straight line programs in the Web Atlas [57]. The language is small and quite constrained, making it easy to write an interpreter for BBOX, but it is powerful enough to implement any of the black box algorithms described in this thesis.

We have written an interpreter for BBOX in GAP, which we used to test the finders and checkers introduced in chapter 1 and described in later chapters. Our interpreter is described in section B.7 below.

B.1 Example: finding an element of order 13 in $L_2(13)$

Before giving the details of BBOX syntax, we will give an example program. The following program finds an element of order 13 in a group $G \simeq L_2(13)$. If none can be found after 100 tries, the algorithm gives up (the probability of this happening in $L_2(13)$ is about 6×10^{-8} , so the algorithm has probably been given the wrong group). If any element is found which does not have an allowable order (given that we are

looking in $L_2(13)$) the algorithm gives an error message.

```

    set X 0
lbl START
    incr X
    rand 1          # Let g_1 be a random element of G
    ord 1 A          # Let A = order(g_1)
    if A notin 1 2 3 6 7 13 then fail
    if A eq 13 then jmp FINISH
    if X eq 100 then timeout
    jmp START

lbl FINISH
    oup 1 1          # Output 1 element, namely g_1

```

B.2 Example: finding standard generators of Fi_{23}

The following is a BBOX program to find standard generators of Fi_{23} . These are elements x and y such that x is in class $2B$, y is in class $3D$ and xy has order 28.

The program falls into two main sections:

- The section starting `lbl SEMISTD` whose purpose is to find elements x and y in the right conjugacy classes
- The section starting `lbl CONJUGATE` whose purpose is to conjugate y until xy has order 28.

Most of the effort is in making sure that the element y is in class $3D$ rather than any other class. This sometimes involves going back to the first step if we discover an element order that we are not expecting.

This program illustrates the most important BBOX language features.

```

# Black box algorithm to find standard generators of Fi23

    set F 0          # Have we found an element of order 2?
    set G 0          # Have we found an element of order 3?
    set V 0          # Timeout counter for "semi-standard" part
    set Y 0          # Timeout counter for conjugating part

lbl SEMISTD

```



```

rand 1
ord 1 A
incr V
if V gt 1000 then timeout
if A notin 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 20 21 22 &
    23 24 26 27 28 30 35 36 39 42 60 then fail
if F eq 0 then
    if A in 20 28 60 then
        div A 2 B
        pwr B 1 2
        set F 1
    endif
endif
endif
if G eq 0 then
    if A in 3 6 9 12 18 then
        div A 3 C      # This is an element of order 3, but it
        pwr C 1 3      # might not be in the right class. We may have
        set G 1        # to revisit this step later.
    endif
endif
endif

if F eq 0 then jmp SEMISTD
if G eq 0 then jmp SEMISTD

set X 0      # Number of times we have tried to prove element is in 3D
set Z 0      # Are we definitely in class 3D?

lbl CONJUGATE
incr Y
if Y gt 1000 then timeout
rand 4
cjr 3 4      # Conjugate y by a random element.
mu 2 3 5     # Compute xy.
ord 5 D      # Find the order of xy.

if D eq 28 then jmp FINISH      # We are done (this proves that y
                                # is in the correct class).

if D notin 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 20 21 22 &
    23 24 26 27 28 30 35 36 39 42 60 then
    fail      # This is not Fi23!

endif

if D in 13 16 17 18 20 21 22 23 24 26 27 28 30 35 36 39 42 60 then
    set Z 1      # y is in class 3D.

endif

if Z eq 0 then
    if D notin 3 6 8 9 10 11 12 13 14 15 16 17 18 20 21 22 23 &
        24 26 27 28 30 35 36 39 42 60 then
        set G 0      # y is definitely
        jmp SEMISTD  # not in class 3D. Try again.
    endif

    incr X
    if X gt 1 then
        set G 0      # We have haven't been able to prove quickly
        jmp SEMISTD  # that y is in class 3D.
                    # We guess that it probably isn't, and start
                    # looking again.
    endif
endif
endif

```

```

    jmp CONJUGATE
lbl FINISH
oup 2 2 3

```

B.3 Language basics

BBOX programs contain one instruction per line. Each instruction begins with a language keyword. The following is a list of BBOX keywords:

```

add, break, call, chcl, chor, cj, cjr, com, cp, decr, div, echo, else,
elseif, endif, fail, false, if, incr, inv, iv, jmp, lbl, mu, mul, nop,
ord, oup, pwr, rand, return, set, sub, timeout, true

```

Each is described in section B.6 below, and many are illustrated in the examples in sections B.1 and B.2.

The end of a line signals the end of one instruction. To avoid some programs having very long lines, an ampersand ‘&’ at the end of a line acts as a line-continuation character (so the line following the ampersand is considered as part of the line containing the ampersand).

Anything following a hash ‘#’ is ignored (this is designed for comments).

B.4 Flow control

Flow proceeds through a BBOX program until a ‘jmp’ or ‘call’ command is reached, when the program moves to the appropriate ‘lbl’ command. At a ‘return’, the program returns to the location of the most recent ‘call’.

Unconditional jumps are unfashionable in computer science [15], but this type of flow control was preferred to more structured commands (such as `do ...while` or `repeat ...until`) because it is frequently important to be able to go back to previous steps if we discover late on that we made an unlucky choice at the beginning of

the algorithm (see for example Algorithm 6.4), and the cleanest way to express this behaviour is with an unconditional jump. Structured commands are also harder to implement, and we felt it was important to keep the language as simple as possible. Black box algorithms are usually short enough that there is no difficulty in understanding the flow through the program.

B.5 Types of identifier

There are two sorts of variables in BBOX: group elements (labelled by $1, 2, \dots$) and counters (labelled by A, B, \dots). Group elements are manipulated by black box commands and oracle commands, and counters are used to conditionally control the flow of the program. As well as variables, we will also need ordinary integers and program labels (used to mark a particular line in the program so that it can be jumped to).

The following notation is used for the different types of identifier in the list of commands below.

- g (possibly with a subscript) is a group element variable. These are labelled by integers $1, 2, 3 \dots$
- c is a counter variable. These are labelled by Roman letters $A, B, C \dots Z$.
- n is an integer.
- λ (possibly with a subscript) is a scalar, *i.e.* a counter variable or an integer.
- L is a program label. These are case-sensitive strings of text.

B.6 Keywords

B.6.1 Black box group commands

<code>oup n $g_1 \dots g_n$</code>	Output a list of n group elements, namely g_1, \dots, g_n
<code>mu g_1 g_2 g_3</code>	Let the group element g_3 be g_1 multiplied by g_2 . This command is not to be confused with <code>mul</code> (which multiplies integers).
<code>iv g_1 g_2</code> (also allow <code>inv</code>)	Let the group element g_2 be the inverse of g_1 .
<code>pwr λ g_1 g_2</code>	Let g_2 be the λ th power of g_1 .
<code>cj g_1 g_2 g_3</code>	Let g_3 be g_1 conjugated by g_2 .
<code>cjr g_1 g_2</code>	Conjugate g_1 by g_2 ‘in place’.
<code>com g_1 g_2 g_3</code>	Let g_3 be the commutator of g_1 and g_2 .
<code>cp g_1 g_2</code>	Copy the element g_1 into g_2 .

B.6.2 Oracle commands

<code>rand g</code>	Let g be a (pseudo-)random element of the group.
<code>ord g c</code>	Let c be the order of the element g .
<code>chor g λ</code>	Check whether element g has order λ . Fail if not.
<code>chcl g \mathcal{C}</code>	Check whether element g could be in conjugacy class \mathcal{C} (typically by taking traces or cycle types). Fail if not. Note that this command is not really a black box command, but was an extension to the language to allow us to perform the conjugacy class tests described in section 11.2.2

B.6.3 Jumping and looping commands

<code>lbl L</code>	Marker for a program label L . This command does not do anything.
<code>jmp L</code>	Jump to the label L .
<code>call L</code>	Jump to the label L and record the current position in the callstack. The command <code>return</code> takes the program back. This is useful for implementing subroutines.
<code>return</code>	Return to the location of the most recent <code>call</code> instruction.

B.6.4 Counter arithmetic

<code>add λ_1 λ_2 c</code>	Let the counter c be $\lambda_1 + \lambda_2$.
<code>sub λ_1 λ_2 c</code>	Let the counter c be $\lambda_1 - \lambda_2$.
<code>mul λ_1 λ_2 c</code>	Let the counter c be $\lambda_1 \times \lambda_2$. This command is not to be confused with <code>mu</code> (which multiplies group elements).
<code>div λ_1 λ_2 c</code>	Let the counter c be the integer part of λ_1 / λ_2 .
<code>mod λ_1 λ_2 c</code>	Let the counter c be the residue of λ_1 modulo λ_2 .
<code>decr c</code>	Decrement the counter c .
<code>incr c</code>	Increment the counter c .
<code>set c λ</code>	Set the counter c to be λ .

B.6.5 Logical commands

There are two forms for logical commands:

- `if predicate then statement`

Single-line form. The *statement* is not allowed to be another `if` statement. If nested logical commands are needed, use the multi-line form.

BBOX predicate	Meaning
$c \text{ eq } \lambda$	$c = \lambda$
$c \text{ noteq } \lambda$	$c \neq \lambda$
$c \text{ in } \lambda_1 \dots \lambda_n$	$c \in \{\lambda_1, \dots, \lambda_n\}$
$c \text{ notin } \lambda_1 \dots \lambda_n$	$c \notin \{\lambda_1, \dots, \lambda_n\}$
$c \text{ lt } \lambda$	$c < \lambda$
$c \text{ leq } \lambda$	$c \leq \lambda$
$c \text{ gt } \lambda$	$c > \lambda$
$c \text{ geq } \lambda$	$c \geq \lambda$

Table B.1: Predicates for the BBOX language

- *if predicate then*

statements

elseif predicate then

statements

...

else

statements

endif

Multi-line form. These can be nested. The *elseif* and *else* clauses are optional.

In both cases, the allowable forms for predicates are given in Table B.1 on page 207.

B.6.6 Terminating commands

<code>true</code>	End the program, and return the boolean value ‘true’ as an answer to a decision problem.
<code>false</code>	End the program, and return the boolean value ‘false’ as an answer to a decision problem.
<code>timeout</code>	Report that the algorithm has spent ‘too long’ on some task. This may suggest that the incorrect group has been given to the algorithm, or it may just be that we were unlucky. This command is intended to end the program, but in the interpreter we implemented (see section B.7), there is an option to continue processing after a time-out.
<code>fail</code>	End the program, and report that the algorithm has determined that the input given is invalid (<i>e.g.</i> the group given is not of the correct isomorphism type). This is a more final mode of failure than that indicated by ‘timeout’.

B.6.7 Debugging commands

<code>echo <i>string</i></code>	Print <i>string</i> to the screen. A counter <i>c</i> preceded by a dollar sign \$ expands to the contents of the counter.
<code>break</code>	Pause the algorithm part way through to allow variables to be examined by hand.

B.7 An interpreter for the BBOX language

Finally, we introduce a simple interpreter for the BBOX language written in GAP.

There are two main commands.

B.7.1 Command: **prepareblackbox**

Syntax: `prepareblackbox(filename)`

This command loads a BBOX program into memory, transforms it into an intermediate form ready for executing and performs a rudimentary syntax check.

If successful, the program returns a structure containing a BBOX program ready for running.

B.7.2 Command: **blackbox**

Syntax: `blackbox(group, program, elements, options)`

This command runs a pre-loaded BBOX program on a given group.

The parameters are:

- *group*: The group on which the algorithm is to be run;
- *program*: The BBOX program (the result of a call to `prepareblackbox`);
- *elements*: A list of elements to become the initial ‘numbered’ elements. These might be a set of group generators for example, or it could just be an empty list.
- *options*: A record containing various options to control the execution of the program. Each component is optional. They have the following meanings:
 - *verbose*: If true, print each instruction before executing. False by default.
 - *quiet*: If true, ignore all echo instructions in the program. False by default.
 - *orderfunction*: A function to replace GAP’s `Order` method. Sometimes useful for large matrix groups, where we substitute a ‘vector order’ method.
 - *classfunction*: A function to test whether a given element could be in a given conjugacy classes (described by its ATLAS name). Typically this involves

looking at traces or cycle types. This function is called when using the `chcl` command.

- *hardtimeout*: If false, continue after reaching a `timeout` instruction (printing a warning). True by default.
- *allowbreaks*: If true, allow `break` instructions, otherwise ignore them. True by default.

If successful, the `blackbox` command returns a record with the following components:

- *gens*: the group elements output by a *oup* command.
- *result*: a true/false value depending on how the program ended. It will be 'true' if the program stopped because it reached a `true` instruction, or if it came to the end of the file. It will be 'false' if the program stopped because it reached a `false` or `timeout` instruction. Otherwise, it will be undefined.
- *multiply, invert, order, random, conjugate, conjugateinplace, commutator*: the number of times instructions of types `mu`, `iv`, `ord/chor`, `rand`, `cj`, `cjr` and `com` were executed.
- *timetaken*: the time taken to complete the black box algorithm in milliseconds.
- *vars*: the contents of the 26 counters.
- *callstack*: the contents of the callstack.

Appendix C

The attached CD-ROM

The representations, black box algorithms and definitions of standard generators referred to in this thesis can be found on the attached CD-ROM. We also include the main programs that we have written: **Split-P** (see section 8.16), **Rho** (see section 10.2.1), **Young** (see section 7.1.1), **Monword** (see section 2.1.26) and a **BBOX** interpreter (see section B.7). At present, the contents of the CD-ROM are also available online at:

<http://web.mat.bham.ac.uk/S.Nickerson/rep/>

The simplest way to browse the CD-ROM is to use a Web browser and open the file `index.html` (found in the root directory). In some operating systems, this index page may open automatically.

Alternatively, one can browse the disc directly. The directory structure and file naming conventions are the same as those for the Web Atlas, so that (for instance) the 90-dimensional representation of J_2 can be found at `spor/J2/gap0/J2G1-Zr90B0.g`. To use this representation in **GAP**, we would use the following syntax:

```
f := ReadAsFunction("J2G1-Zr90B0.g");;
gens := f().generators;;
G := Group(gens);
```

List of References

- [1] R. W. Barraclough and R. A. Wilson. Conjugacy class representatives in the Monster. Preprint, 2005.
- [2] Robert Beals, Charles R. Leedham-Green, Alice C. Niemeyer, Cheryl E. Praeger, and Ákos Seress. A black-box group algorithm for recognizing finite symmetric and alternating groups. I. *Trans. Amer. Math. Soc.*, 355(5):2097–2113 (electronic), 2003.
- [3] Áron Bereczky. Maximal overgroups of Singer elements in classical groups. *J. Algebra*, 234(1):187–206, 2000.
- [4] W. Bosma and J. Cannon. *The MAGMA Handbook*.
- [5] Sergey Bratus and Igor Pak. Fast constructive recognition of a black box group isomorphic to S_n or A_n using Goldbach’s conjecture. *J. Symbolic Comput.*, 29:33–57, 2000.
- [6] John N. Bray and Robert T. Curtis. Monomial modular representations and symmetric generation of the Harada-Norton group. *J. Algebra*, 268(2):723–743, 2003.
- [7] Thomas Breuer and Götz Pfeiffer. Finding possible permutation characters. *J. Symbolic Comput.*, 26(3):343–354, 1998.
- [8] W. Burnside. *Theory of groups of finite order*. Dover Publications Inc., New York, 1955. 2d ed.
- [9] R. D. Carmichael. Abstract definitions of the symmetric and alternating groups and certain other permutation groups. *Quart. J. Math.*, 49:226–270, 1923.
- [10] Frank Celler and C. R. Leedham-Green. Calculating the order of an invertible matrix. In *Groups and computation, II* (New Brunswick, NJ, 1995), volume 28 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 55–60. Amer. Math. Soc., Providence, RI, 1997.
- [11] J. H. Conway, R. T. Curtis, S. P. Norton, R. A. Parker, and R. A. Wilson. *An ATLAS of Finite Groups*. Oxford University Press, 1985.

- [12] Rodney Cruz and Pablo M. Salzberg. A shift and cut GCD algorithm. In *Proceedings of the Twenty-fifth Southeastern International Conference on Combinatorics, Graph Theory and Computing* (Boca Raton, FL, 1994), volume 103, pages 65–76, 1994.
- [13] Vahid Dabbaghian-Abdoly. *An algorithm to construct representations of finite groups*. PhD thesis, Carleton University, 2003.
- [14] P. Deligne and G. Lusztig. Representations of reductive groups over finite fields. *Ann. of Math. (2)*, 103(1):103–161, 1976.
- [15] Edsger W. Dijkstra. Go to statement considered harmful. *Comm. ACM*, 11(3):147–8, March 1968.
- [16] John D. Dixon. Constructing representations of finite groups. In *Groups and computation* (New Brunswick, NJ, 1991), volume 11 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 105–112. Amer. Math. Soc., Providence, RI, 1993.
- [17] Larry Dornhoff. *Group Representation Theory (Part A)*. Marcel Denker, 1971.
- [18] Robert E. Dressler. A stronger Bertrand’s postulate with an application to partitions. *Proc. Amer. Math. Soc.*, 33:226–228, 1972.
- [19] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.4*, 2004. (<http://www.gap-system.org>).
- [20] M. Geck, G. Hiss, F. Lübeck, G. Malle, and G. Pfeiffer. CHEVIE – A system for computing and processing generic character tables for finite groups of Lie type, Weyl groups and Hecke algebras. *Appl. Algebra Engrg. Comm. Comput.*, 7:175–210, 1996.
- [21] Gerhard Hiss and Gunter Malle. Low-dimensional representations of quasi-simple groups. *LMS J. Comput. Math.*, 4:22–63 (electronic), 2001.
- [22] Gerhard Hiss and Gunter Malle. Corrigenda: “Low-dimensional representations of quasi-simple groups” [LMS J. Comput. Math. 4 (2001), 22–63; MR 2002b:20015]. *LMS J. Comput. Math.*, 5:95–126 (electronic), 2002.
- [23] Derek F. Holt. The Meataxe as a tool in computational group theory. In *The Atlas of Finite Groups: Ten Years On* (Birmingham, 1995), volume 249 of *London Math. Soc. Lecture Note Ser.*, pages 74–81. Cambridge Univ. Press, Cambridge, 1998.
- [24] B. Huppert. *Endliche Gruppen. I*. Die Grundlehren der Mathematischen Wissenschaften, Band 134. Springer-Verlag, Berlin, 1967.
- [25] I. Martin Isaacs. *Character theory of finite groups*. Dover Publications Inc., New York, 1994. Corrected reprint of the 1976 original [Academic Press, New York].

- [26] Gordon James and Martin Liebeck. *Representations and characters of groups*. Cambridge Mathematical Textbooks. Cambridge University Press, Cambridge, 1993.
- [27] Christopher Jansen, Klaus Lux, Richard Parker, and Robert Wilson. *An Atlas of Brauer Characters*. Number 11 in LMS Monographs. Oxford Science Publications, 1995.
- [28] Brian W. Kernighan and Dennis M. Ritchie. *The C Programming Language*. Prentice Hall, 2nd edition, 1988.
- [29] Peter Kleidman and Martin Liebeck. *The Subgroup Structure of the Finite Classical Groups*. Number 129 in London Mathematical Society Lecture Note Series. Cambridge University Press, 1990.
- [30] Torleiv Kløve. Sums of distinct primes. *Nordisk Mat. Tidskr.*, 21:138–140, 1973.
- [31] Stephen Linton, Richard Parker, Peter Walsh, and Robert Wilson. Computer construction of the Monster. *J. Group Theory*, 1(4):307–337, 1998.
- [32] Richard S. Margolin. A geometry for M_{24} . *J. Algebra*, 156(2):370–384, 1993.
- [33] S. J. Nickerson and R. A. Wilson. Semi-presentations for the sporadic simple groups. *Experiment. Math.*, 14(3):359–371, 2005.
- [34] Simon Nickerson. Maximal subgroups of the sporadic almost-simple groups. MPhil (Qual) thesis, The University of Birmingham, 2004.
- [35] Simon Norton. *F and other simple groups*. PhD thesis, University of Cambridge, 1975.
- [36] R. A. Parker. The computer calculation of modular characters (the meat-axe). In *Computational group theory (Durham, 1982)*, pages 267–274. Academic Press, London, 1984.
- [37] Richard A. Parker. An integral meataxe. In *The Atlas of Finite Groups: Ten Years On (Birmingham, 1995)*, volume 249 of *London Math. Soc. Lecture Note Ser.*, pages 215–228. Cambridge Univ. Press, Cambridge, 1998.
- [38] Martin Pergler. Complex representations of $gl(2, q)$. *C. R. Math. Rep. Acad. Sci. Canada*, 17:207–212, 1995.
- [39] Ilya Piatetski-Shapiro. *Complex representations of $GL(2, K)$ for finite fields K* , volume 16 of *Contemporary Mathematics*. American Mathematical Society, Providence, R.I., 1983.
- [40] Wilhelm Plesken and Bernd Souvignier. Constructing rational representations of finite groups. *Experiment. Math.*, 5(1):39–47, 1996.

- [41] Michael Ringe. *The C MeatAxe Release 1.5*. Lehrstuhl D für Mathematik, Rheinisch Westfälische Technische Hochschule, Aachen, Germany, 1993.
- [42] Stephen J. F. Rogers. *Representations of finite simple groups over fields of characteristic zero*. PhD thesis, University of Birmingham, 1997.
- [43] Bruce E. Sagan. *The symmetric group*. The Wadsworth & Brooks/Cole Mathematics Series. Wadsworth & Brooks/Cole Advanced Books & Software, Pacific Grove, CA, 1991. Representations, combinatorial algorithms, and symmetric functions.
- [44] A. Schönhage and V. Strassen. Schnelle Multiplikation grosser Zahlen. *Computing (Arch. Elektron. Rechnen)*, 7:281–292, 1971.
- [45] I. Schur. Untersuchungen über die Darstellungen der endlichen Gruppen durch gebrochene lineare Substitutionen. *J. Reine Angew. Math.*, 132:85–137, 1907.
- [46] Ákos Seress. *Permutation Group Algorithms*. Number 152 in Cambridge Tracts in Mathematics. Cambridge University Press, 2003.
- [47] N. J. A. Sloane. The on-line encyclopedia of integer sequences. Published electronically at <http://www.research.att.com/~njas/sequences/>, 2005.
- [48] P. E. Smith. A simple subgroup of M_n and $E_8(3)$. *Bull. London Math. Soc.*, 8(2):161–165, 1976.
- [49] Jonathan Sorenson. Two fast GCD algorithms. *J. Algorithms*, 16(1):110–144, 1994.
- [50] Bhama Srinivasan. The characters of the finite symplectic group $\mathrm{Sp}(4, q)$. *Trans. Amer. Math. Soc.*, 131:488–525, 1968.
- [51] Fernando Szechtman. Weil representations of the symplectic group. *J. Algebra*, 208(2):662–686, 1998.
- [52] Shun’ichi Tanaka. Construction and classification of irreducible representations of special linear group of the second order over a finite field. *Osaka J. Math.*, 4:65–84, 1967.
- [53] University of Waterloo. *Maple V, release 3*.
- [54] Joachim von zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge University Press, New York, 1999.
- [55] Larry Wall. *The Perl Programming Language*. <http://www.perl.com>.
- [56] Xinmao Wang and Victor Y. Pan. Acceleration of Euclidean algorithm and rational number reconstruction. *SIAM J. Comput.*, 32(2):548–556 (electronic), 2003.

- [57] Robert Wilson, Peter Walsh, Jonathan Tripp, Ibrahim Suleiman, Stephen Rogers, Richard Parker, Simon Norton, Simon Nickerson, Steve Linton, John Bray, and Rachel Abbott. Atlas of Finite Group Representations. Available online at <http://web.mat.bham.ac.uk/atlas/>.
- [58] Robert A. Wilson. Standard generators for sporadic simple groups. *J. Algebra*, 184(2):505–515, 1996.
- [59] Robert A. Wilson. The Monster is a Hurwitz group. *J. Group Theory*, 4(4):367–374, 2001.